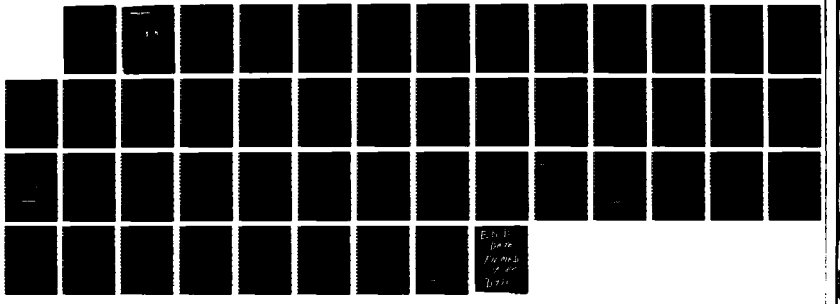
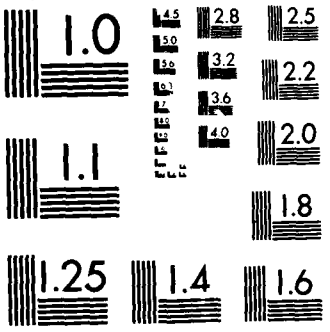
AD-A195 526 KALMAN FILTER ARCHITECTURES FOR BATTLE MANAGEMENT PHASE 1/1
1(U) SYSTOLIC SYSTEMS INC SAN JOSE CA D SATTERFIELD
JAN 87 DASG60-86-C-0056

UNCLASSIFIED F/G 12/5 NL





SYSTOLIC SYSTEMS
INCORPORATED

File

Copy 1

KALMAN FILTER ARCHITECTURES FOR BATTLE MANAGEMENT

SBIR PHASE I

CONTRACT DASG60-86-C-0056

Prepared for:

DEPARTMENT OF THE ARMY
OFFICE OF THE CHIEF OF STAFF
U.S. ARMY STRATEGIC DEFENSE COMMAND - HUNTSVILLE
P. O. BOX 1500
HUNTSVILLE, ALABAMA 35807-3801
ATTN: DR. DOYCE SATTERFIELD

FINAL REPORT 420102, JANUARY 1987

240 NORTH FIRST STREET • SAN JOSE, CALIFORNIA 95131-2310 • 408/435-1760 • INT TELEX 184817 • FAX 408/287-4535

FOREWORD

This report develops a comprehensive theory of parallel Kalman filtering based on a unique decoupling principle permitting the predictor and corrector equations in the filter to be computed in parallel. Highly parallel algorithms and systolic architectures for efficiently implementing these advanced filtering techniques are presented. The application of these methods to Strategic Defense Initiative (SDI) target tracking problems is also described. → top 6-3

This research was sponsored by the U. S. Army Strategic Defense Command and conducted under U. S. Army contract number DASG60-86-C-0056. Dr. Doyce Satterfield was the program manager at the U. S. Army Strategic Defense Command - Huntsville.

Dr. Richard H. Travassos was the Principal Investigator at Systolic Systems, Inc. Other members of the technical staff who contributed to this project include Gary Lee and Larry Hubbart.

TABLE OF CONTENTS

SECTION 1

INTRODUCTION

1.1 BACKGROUND

In navigation equipment, such as the GPS, a Kalman filter acts to smooth data when the unit is unaided, or as an estimating filter when inertial data are accepted. The need to integrate multiple sensors results in a hybrid system with extremely large computational requirements for real-time applications. Often the hybrid system takes the form of a "cascaded" filter to ease the computational burden. Sensor data integration is often difficult due to correlating the time of the measurements and the different measurement rates of the sensors. In this project, parallel Kalman filter architectures are optimized for hybrid systems consisting of multiple sensors to achieve improved performance. The computational advantage of parallel processing minimizes measurement time sensitivity and data transfer over the bus. Thus, multi-rate filtering is attainable via a unique decoupling of the predictor and corrector equations in the filter while maintaining optimal estimation. The parallel processing techniques can be applied to real-time navigation, target tracking and scene analysis.

With recent developments in advanced sensors, a severe load has been placed upon real-time signal processing systems to process large amounts of data. Although the technology for implementing advanced sensors already exists, the actual implementation depends strongly on the ability to develop real-time signal processing hardware to process the data. Thus, to meet the exceptionally high throughput requirements in DoD signal processing applications, considerable attention must be given to the development of highly parallel (or systolic) signal processing architectures. Because signal processing architectures tend to be problem dependent to achieve the necessary computational requirements, this project develops systolic signal processor architectures which are well suited for recursive filtering, target tracking, image processing and signal processing. The parallel architectures are based on mapping the widely-used Kalman filter equations onto a generalized systolic

array for linear and nonlinear parallel computation. Although this was originally developed by Travassos (1982) for linear estimation on two (2) processors, the extension of the method to n processors is the thrust of this project.

To illustrate the need to extend the method for SDI sensor track processing via a Kalman filter consider the simplest case for linear filtering (nonlinear filtering is even more computationally demanding). The total number of arithmetic operations that must be computed in the Kalman filter algorithm can be counted and multiplied by different multiplier and adder speeds. For the Kalman filter algorithm given in Table 1, the total number of multiplications, additions and divisions is given by:

(n*n + 2n - 1) additions, (2n*n + 4n + 1) multiplications, and 1 division.

Table 1: Standard Kalman Filter Algorithm*

Kalman* Gain	$K_k = P_k(+)H_k^T [H_k^T P_k(+)H_k^T + 1]^{-1}$
Filter Update	$\hat{x}_k(+) = \hat{x}_k(-) + K_k [v_k - H_k \hat{x}_k(-)]$ $= [I - K_k H_k] \hat{x}_k(-) + K_k v_k$
Covariance Update	$P_k(+) = [I - K_k H_k] P_k(-)$
Measurement Update	$Y_{k+1} = H_{k+1} \hat{x}_k(+)$

*Note that when scalar measurements are processed, the inverse operation reduces to a division operation.

Therefore, the overall execution time needed to update the Kalman filter algorithm in

$$t = (n*n + 2n - 1)\ t_a + (2n*n + 4n + 1)\ t_m + t_d$$

where

t_a is the addition time, t_m is the multiplication time, and t_d is the division time.

For example, if t_a = 125 nsec, t_m = 200 nsec and t_d = 1000 nsec, one pass through the Kalman filter with n = 9 states requires 53 μsec. This corresponds to 1/0.53 μsec = 18,868 updates per second to process one track using state-of-the-art 32-bit floating-point VLSI chips assuming 100% efficiency. Typically, however, only 10 to 30% of peak performance is achieved in practice. Therefore, one target track may be updated at a 4,000 updates per second rate. 1000 targets could be updated at a 4 Hz rate and 10,000 targets at .4 Hz rate (every 2.5 seconds). For nonlinear filtering, typical of SDI target tracking problems (see Section 2), 64-bit precision and the need to compute trigonometric functions for coordinate transformations can slow computations down by 1 or perhaps 2 orders of magnitude (10x to 100x). Since it is well known that the Kalman filtering must be performed using floating-point arithmetic to avoid stability problems, the only viable method to gain back the throughput for SDI filtering problems using an extended Kalman filter is with parallel processing. Optical processing is fast but optical fixed-point can cause stability problems with the Kalman filter. This report, therefore, extends a systolic architecture approach for rapidly implementing parallel Kalman filters with 32/64-bit floating-point electronic technology for several SDI applications.

In this report, systolic array concepts are used to develop efficient architectures for implementing the bank of Kalman filters shown in Figure 1-1 needed by the recursive maximum-likelihood estimator.

A system of 32x32 = 1024 processors based on the methods in this report will provide 32/64-bit, IEEE standard computations at speeds approaching 1000x faster than today's technology. Note that decoupling the problem to run on several processing elements is required not just high-speed arithmetic units.

1.2 SUMMARY OF THE APPROACH

The Phase I plan was to examine the feasibility of extending the 2 processor parallel Kalman filter algorithms and architectures from two (2) to n (even) processors. To show feasibility, the method was extended from two (2) to four (4) processors and then to n (even) processors. The optimality of the 2

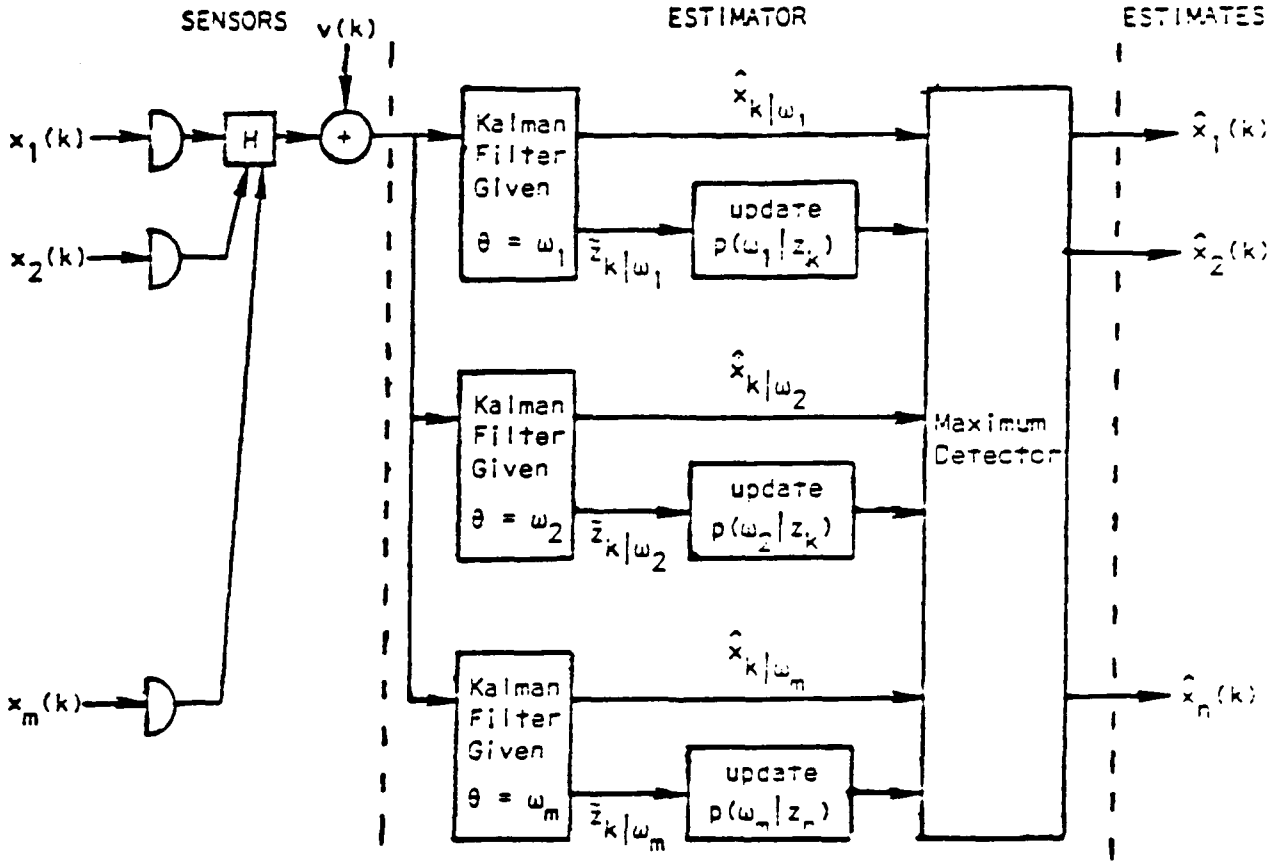


FIGURE 1-1 Maximum-Likelihood Estimation of Multiple Targets

processor case was also researched. Parallel architectures have also been developed to implement the parallel Kalman filter algorithms. A summary of the Phase I technical approach is given in Figure 1-2.

Our Phase I technical approach covered the following areas:

- Defining a realistic SDI sensor estimation problem
- Examining the class of computations required for solving SDI target tracking problems
- Restructuring the Kalman filter equations for parallel processing
- Expanding previously-developed systolic Kalman filter algorithms and architectures from 2 processor to n (even) processors
- Investigating the feasibility of implementing the parallel Kalman filter architectures in hardware taking into account wordlength, sampling rate, memory and reliability issues

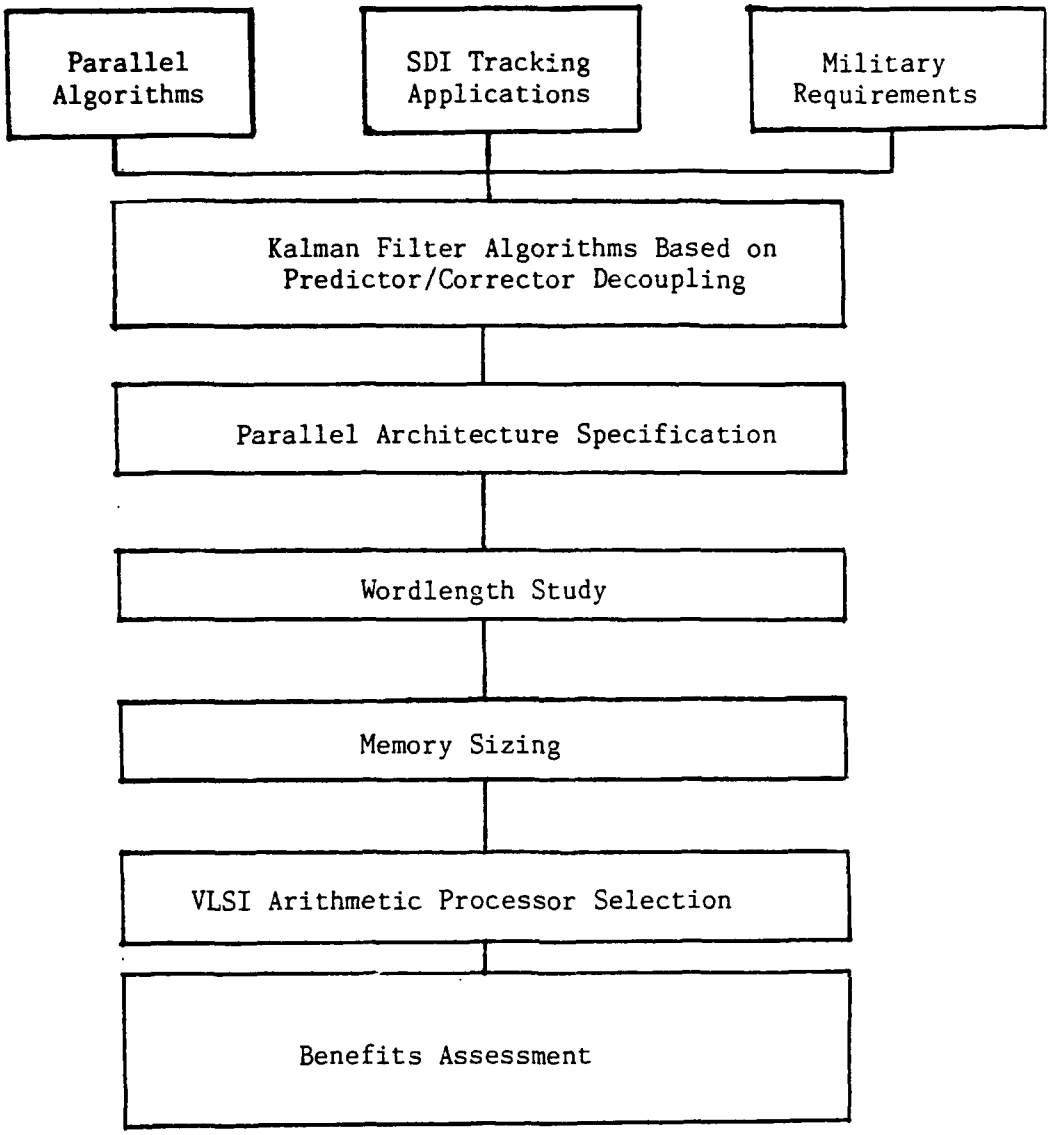


Figure 1-2 Phase I Technical Approach

The major payoffs of the research included:

- Solving SDI problems which could not be solved otherwise by providing three (3) orders of magnitude improvement (1000x) in computational speed over existing array processor-based Kalman filter implementations
- Constructing a generic Battle Management Testbed Facility that can be used to validate new parallel Kalman filter algorithms as they become available. The testbed permitted parallel algorithms and architectures to be rapidly evaluated, speeding up the deployment of new designs in SDI systems
- Computational throughput was measured explicitly , rather than estimated, taking into account processor-to-processor and processor-to-memory communications in the testbed.

1.3 RESULTS

The major results of the Phase I feasibility study can be summarized as follows:

- The "optimality" of the parallel Kalman filter has been proven analytically by showing that the dual (2) processor parallel Kalman filter is "mathematically equivalent" to the standard (sequential) Kalman filter. The dual (2) processor parallel Kalman filter, however, executes twice as fast as the standard filter since the predictor and corrector in the parallel Kalman filter can be computed in parallel.

- The parallel Kalman filter for linear estimation problems has been extended from 2 to n (even) processors.

- Parallel architectures have also been developed to rapidly implement the linear parallel Kalman filter methods.

- The wordlength, memory and VLSI arithmetic processor selections have been examined and documented.

- In addition to the above, the linear parallel Kalman filter has been extended for nonlinear track processing. The extended, nonlinear parallel Kalman filter equations have been developed for the 2 (and 4) processor case under Phase I. Under Phase II, the nonlinear extended parallel Kalman filter can be further decoupled and generalized for n (even) processors.

1.4 REPORT SUMMARY

The remainder of this report is organized as follows. The decoupling of the parallel Kalman filter (PKF) for linear estimation is presented in Section 2. The optimality of the 2 processor linear PKF is proven analytically in Section 2. The generalization of the two (2) processor linear PKF equations to execute on n (even) processors is also developed in Section 2. Nonlinear Kalman filtering via an extended parallel Kalman filter is presented in Section 3. Parallel architectures for efficiently implementing the PFK equations can be found in Sections 2 and 3. The wordlength, memory sizing and VLSI arithmetic processor selection is given in Section 4. The SDI target tracking problem is formulated in Section 5 for demonstrating the utility of the PKF methods. Conclusions and recommendations for future work are presented in Section 6.

SECTION 2

LINEAR PARALLEL KALMAN FILTER ALGORITHMS AND ARCHITECTURES BASED ON THE DECOUPLING PRINCIPLE

The Kalman filter has been successfully applied to many signal processing applications including target prediction, target tracking, radar signal processing, on-board calibration of intertial systems and in-flight estimation of aircraft stability and control derivatives. The applicability of the Kalman filter to real-time processing problems is generally limited, however, by the filter's relative computational complexity. In particular, the number of arithmetic operations required for implementing the Kalman filter with a state variable grows as $0(n^2)$ for the time update and as $0(n^3)$ for the covariance update. In general, real-time filtering cannot be performed on large scale problems using a uniprocessor architecture because serious processing lags can result (9).

The Kalman filter can be extended to a much greater class of problems by using parallel processing concepts. Full utilization of parallelism can be obtained through insight in the structure of the problem and decoupling of arithmetic processes to permit concurrent processing.

To speed up Kalman filter computations, parallel processing is performed at two levels: (1) the predictor and corrector equations of the Kalman filter are decoupled so that the predictor and corrector can be computed on separate processors, and (2) the measurement data are pipelined into each processor. Therefore, both multiprocessing and pipelining are considered to achieve large improvements in computational speed.

2.1 DECOUPLING THE TIME AND MEASUREMENT UPDATE

The Kalman filter equations in Table 1-1 can be written in predictor-corrector form as follows:

Predictor

$$\hat{x}_k(-) = \phi_{k-1}\hat{x}_{k-1}(+) \tag{2.1}$$

$$P_k(-) = \phi_{k-1}P_k(+)\phi_{k-1}^T \tag{2.2}$$

$$\text{Corrector} \begin{cases} \hat{x}_k(+) = \hat{x}_k(-) + K_k(z_k - H_k\hat{x}_k(-)) & (2.3) \\ P_k(+) = (I - K_kH_k)\, P_k(-) & (2.4) \end{cases}$$

where the Kalman gain is

$$K_k = P_k(-)\, H_k^T(H_kP_k(-)\, H_k^T + R_k)^{-1} \tag{2.5}$$

and ϕ_k = state transition matrix. (2.6)

Note that Eqs. (2.1) to (2.5) are inherently sequential since the temporal updates (predictor equations) must be evaluated before the observation updates (corrector equations). From a computational point of view, this is not desirable since to evaluate the corrector a uniprocessor must wait until the predictor has been evaluated. To avoid this difficulty, the predictor-corrector equations can be decoupled to obtain a parallel Kalman filter (PKF).

2.2 DECOUPLING THE STATE UPDATE (Travassos, 1982)

The decoupling of the predictor and corrector is achieved by forcing the corrector to lag the predictor by one time step as follows:

$$\text{Predictor:} \quad \hat{x}_{k+1}(-) = \phi_k\phi_{k-1}\hat{x}_{k-1}(+)\ , \tag{2.7}$$

$$\text{Corrector:} \quad \hat{x}_k(+) = \hat{x}_k(-) + K_k(z_k - H_k\hat{x}_k(-))\ . \tag{2.8}$$

2.3 DECOUPLING THE COVARIANCE UPDATE (Travassos, 1982)

Let the covariance of the estimation error before and after a measurement update be denoted by:

$$P_{k+1}(-) = E\tilde{x}_{k+1}(-)\, \tilde{x}_{k+1}^T(-)\ , \tag{2.9}$$

$$P_k(+) = E\tilde{x}_k(+)\, \tilde{x}_k^T(+)\ , \tag{2.10}$$

where

$$\tilde{x}_{k+1}(-) \triangleq \hat{x}_{k+1}(-) - x_{k+1}\ , \tag{2.11}$$

$$\tilde{x}_k(+) \triangleq \hat{x}_k(+) - x_k\ . \tag{2.12}$$

By direct computation, it can be shown that the covariance of the estimation error before the update is given by:

$$P_{k+1}(-) = \phi_k\phi_{k-1}(+)\, P_{k-1}\phi_{k-1}^T\phi_k^T \tag{2.13}$$

Because the form of Eq. (2.8) is the same as Eq. (2.9), the covariance of the

estimation error after a measurement update in the PKF is given by:

$$P_k(+) = (I - K_k H_k)\, P_k(-) \ , \tag{2.14}$$

where

$$K_k = P_k(-)\, H_k^T (H_k P_k(-)\, H_k^T + R_k)^{-1} \tag{2.15}$$

2.4 SUMMARY OF THE DECOUPLED PKF EQUATIONS FOR TWO PROCESSORS (Travassos, 1982)

Because the predictor and corrector equations in the Kalman filter can be decoupled, computations can be performed simultaneously on two separate processors, one processor for the predictor equations and one processor for the corrector equations. In summary, the parallel Kalman filter (PKF) equations are:

Predictor

$$\hat{x}_{k+1}(-) = \phi_k \phi_{k-1} \hat{x}_{k-1}(+) \tag{2.16}$$

$$P_{k+1}(-) = \phi_k \phi_{k-1} P_{k-1}(+)\, \phi_{k-1}^T \phi_k^T \tag{2.17}$$

Corrector

$$\hat{x}_k(+) = \hat{x}_k(-) + K_k (z_k - H_k \hat{x}_k(-)) \tag{2.18}$$

$$P_k(+) = (I - K_k H_k)\, P_k(-) \tag{2.19}$$

Kalman Gain

$$K_k = P_k(-)\, H_k^T (H_k P_k(-)\, H_k^T + R_k)^{-1} \tag{2.20}$$

Although these results can be applied to a "linearized" model of the sensor track processing problem, we propose to extend these results to work with the target's nonlinear equations of motion directly. Using the nonlinear equations of motion will provide more accurate tracking (see Section 3.1) for the nonlinear parallel Kalman filter equations). Both linear and nonlinear parallel Kalman filter techniques can be applied.

2.5 OPTIMALITY OF THE PARALLEL KALMAN FILTER

To show that the PKF is optimal (in the same sense as the standard Kalman filter), the optimality proof will be separated into two parts. First we shall show that the PKF generates the same state update as the SKF. Secondly, the covariance update in the PKF is shown to be equivalent to the covariance update in the PKF. Combining these results shows that the PKF is optimal since both the state and covariance updates are "mathematically equivalent" to the SKF.

The State Update:

For the SKF, substituting Eq. (2.9) into Eq. (2.7) evaluated at time k+1, we have:

$$\hat{x}_{k+1}(-) = \phi_k \hat{x}_k(-) + \phi_k K_k (z_k - H_k \hat{x}_k(-)) \tag{2.21}$$

where

$$K_k = P_k(-)\, H_k^T (H_k P_k(-)\, H_k^T + R_k)^{-1} \tag{2.22}$$

Note that the innovations sequence is given by:

$$\tilde{z}_{k/k+1} = (z_k - H_k \hat{x}_k(-)) \qquad k = 0, 1, 2, \ldots \tag{2.23}$$

Substituting Eq. (2.7) back into Eq. (2.21), we have

$$\hat{x}_{k+1}(-) = \underbrace{\phi_k \phi_{k-1} \hat{x}_{k-1}(+)}_{\text{transition update}} + \underbrace{\phi_k K_k (z_k - H_k \hat{x}_k(-))}_{\text{innovations update}} \tag{2.24}$$

Thus, the state update in the SKF consists of two parts - the transition update and innovations update. Our objective was to show that the PKF state update is equivalent to Eq. (2.24). To do this, we needed to prove the following lemma:

Lemma #1

For the PKF defined by Eqs. (2.16) to (2.20),

IF $\hat{x}_k(+) \triangleq \phi_{k-1} \hat{x}_{k-1}(+)$ (2.25)

THEN $\hat{x}_{k-1}(-) = \hat{x}_{k-1}(+)$ (2.26)

AND $\hat{x}_k(-) = \phi_{k-1} \hat{x}_{k-1}(-)$ (2.27)

PROOF: Evaluating Eq. (2.16) at time k, we have

$$\hat{x}_k(-) = \phi_{k-1} \phi_{k-2} \hat{x}_{k-2}(+) \tag{2.28}$$

But $\phi_{k-2} \hat{x}_{k-2}(+) = \hat{x}_{k-1}(+)$ by Eq. (2.25) evaluated at time k-1. Evaluating Eq. (2.28) at time k-1 results in

$$\hat{x}_{k-1}(-) = \phi_{k-2} \hat{x}_{k-2}(+) = \hat{x}_{k-1}(+) \tag{2.29}$$

by virtue of Eq. (2.25). Thus,

$$\hat{x}_{k-1}(-) = \hat{x}_{k-1}(+) \ . \tag{2.30}$$

The next part of the proof is to show that $\hat{x}_k(-) = \phi_{k-1}\hat{x}_{k-1}(-)$ in the PKF. Evaluating Eq. (2.16) at time k+1, we have:

$$\hat{x}_{k+1}(-) = \phi_k\phi_{k-1}\hat{x}_{k-1}(+) = \phi_k\hat{x}_k(+) \tag{2.31}$$

or at time k

$$\hat{x}_k(-) = \phi_{k-1}\hat{x}_{k-1}(+) \tag{2.32}$$

But by Eq. (2-20) we can conclude the proof as follows:

$$\hat{x}_k(-) = \phi_{k-1}\hat{x}_{k-1}(-) \quad \blacksquare \tag{2.33}$$

Now we can use the results of Lemma #1 to show that the PKF state update is "mathematically equivalent" to the SKF state update.

Theorem 1:

Because the transition state update

$$\phi_k\phi_{k-1}\hat{x}_{k-1}(+) = \phi_k\phi_{k-1}\hat{x}_{k-1}(-) \tag{2.34}$$

and innovations state update

$$\phi_{k-1}K_{k-1}(z_{k-1} - H_{k-1}\hat{x}_{k-1}(+)) = K_k(z_k - H_k\hat{x}_k(-)) \tag{2.35}$$

are mathematically equivalent, the PKF and SKF state updates are mathematically equivalent.

Proof: The transition update is easy using Lemma #1, Eq. (2.26) since:

$$\phi_k\phi_{k-1}\hat{x}_{k-1}(-) = \phi_k\phi_{k-1}\hat{x}_{k-1}(+) \tag{2.36}$$

Pre-multiplying the innovations in the PKF state update, we have:

$$\phi_{k-1}K_{k-1}(z_{k-1} - H_{k-1}\hat{x}_{k-1}(+)) = \phi_{k-1}(\hat{x}_{k-1}(+) - \hat{x}_{k-1}(-)) \tag{2.37}$$

by Eq. (2.18) evaluated at time k-1. Expanding Eq. (2.37) results in

$$= \phi_{k-1}\hat{x}_{k-1}(+) - \phi_{k-1}\hat{x}_{k-1}(-) \tag{2.38}$$

$$= \hat{x}_k(+) - \hat{x}_k(-) \tag{2.39}$$

by Eqs. (2.25) and (2.27) in Lemma #1. But $\hat{x}_k(+) - \hat{x}_k(-) = K_k(z_k - H_k\hat{x}_k(-))$ using Eq. (2.18) in the PKF. Thus, it can be concluded that

$$\phi_{k-1}K_{k-1}(z_{k-1} - H_{k-1}\hat{x}_{k-1}(+)) = K_k(z_k - H_k\hat{x}_k(-)) \quad \blacksquare \tag{2.40}$$

as desired.

Now that we have shown that the PKF and SKF state updates are "mathematically equivalent," we turn our attention to the covariance update.

The Covariance Update:

From Eqs. (2.7) to (2.11), the covariance update in the SKF can be summarized as follows:

$$P_k(-) = \phi_{k-1} P_{k-1}(+) \phi_{k-1}^T \tag{2.41}$$

$$P_k(+) = (I - K_k H_k)\, P_k(-) = P_k(-) - K_k H_k P_k(-) \tag{2.42}$$

Replacing $P_{k-1}(+)$ in Eq. (2.41 evaluated at time k+1 with Eq. (2.42) we have:

$$P_{k+1}(+) = \phi_k P_k(-)\, \phi_k^T - \phi_k K_k H_k P_k(-)\, \phi_k^T \tag{2.43}$$

Now substituting Eq. (2.41) evaluated at time k into Eq. (2.) results in

$$P_{k+1}(+) = \underbrace{\phi_k \phi_{k-1} P_{k-1}(+)\, \phi_{k-1}^T \phi_k^T}_{\text{transition update}} - \underbrace{\phi_k K_k H_k P_k(-)\, \phi_k^T}_{\text{gain update}} \tag{2.44}$$

which summarizes the complete SKF covariance update. As before, our objective is to show that the PKF covariance update is "mathematically equivalent" to Eq. (2.44). Another lemma is useful in proving this result.

Lemma #2

For the PKF defined by Eqs. (2.16) to (2.20),

IF $P_k(+) \triangleq \phi_{k-1} P_{k-1}(+)\, \phi_{k-1}^T$ (2.45)

THEN $P_{k-1}(-) = P_{k-1}(+)$ (2.46)

AND $P_k(-) = \phi_{k-1} P_{k-1}(+)\, \phi_{k-1}^T$ (2.47)

PROOF: Evaluating Eq. (2.17) at time k, we have

$$P_k(-) = \phi_{k-1} \phi_{k-2} P_{k-2}(+)\, \phi_{k-2}^T \phi_{k-1}^T \tag{2.48}$$

But by Eq. (2.45) evaluated at time k-1, Eq. (2.48) can be written as follows:

$$P_k(-) = \phi_{k-1} P_{k-1}(+)\, \phi_{k-1}^T \quad \text{which proves Eq. (2.47)} \tag{2.49}$$

Evaluating Eq. (2.49) at time k-1, results in

$$P_{k-1}(-) = \phi_{k-2} P_{k-2}(+)\, \phi_{k-2}^T = P_{k-1}(+) \tag{2.50}$$

using Eq. (2.45) evaluated at time k-1. Hence, $P_{k-1}(-) = P_{k-1}(+)$ as desired.

Now we can use the results of Lemma #2 to show that the PKF covariance update is "mathematically equivalent" to the SKF covariance update.

Theorem #2:

Because the transition covariance update

$$\phi_k\phi_{k-1}P_{k-1}(-)\ \phi_{k-1}^T\phi_k^T = \phi_k\phi_{k-1}P_{k-1}(+)\ \phi_{k-1}^T\phi_k^T$$

and the gain covariance update

$$\phi_{k-1}K_{k-1}H_{k-1}P_{k-1}(-)\ \phi_{k-1}^T = K_kH_kP_k(-)$$

are mathematically equivalent, the PKF and SKF covarinace updates are mathematically equivalent.

PROOF: To begin, substitute the PKF covariance update Eq. (2,19) evaluated at time k-1 into Eq. (2.17) to obtain:

$$P_{k+1}(-) = \phi_k\phi_{k-1}(I - K_{k-1}H_{k-1})\ P_{k-1}(-)\ \phi_{k-1}^T\phi_k^T$$

$$= \underbrace{\phi_k\phi_{k-1}P_{k-1}(-)\ \phi_{k-1}^T\phi_k^T}_{\text{transition covariance}} - \phi_k\underbrace{(\phi_{k-1}K_{k-1}H_{k-1}P_{k-1}(-)\ \phi_{k-1}^T)}_{\text{gain covariance}}\ \phi_k^T \quad (2.51)$$

Note that if we can prove that

$$\phi_k\phi_{k-1}P_{k-1}(-)\ \phi_{k-1}^T\phi_k^T = \phi_k\phi_{k-1}P_{k-1}(+)\ \phi_{k-1}^T\phi_k^T \quad (2.52)$$

and

$$\phi_{k-1}K_{k-1}H_{k-1}P_{k-1}(-)\ \phi_{k-1}^T = K_kH_kP_k(-) \quad (2.53)$$

then the PKF update in Eq. (2.51) will be the same (i.e., equivalent) to the SKF update in Eq. (2.44). Therefore, the PKF would be optimal since both the state and covariance updates of PKF are "mathematically equivalent" to the SKF. Eq. (2.52) is easy to prove due to Eq. (2.46) in Lemma #2. Now to prove Eq. (2.53). To do this, note from the PKF Eq. (2.19) evaluated at time k-1 we can write:

$$K_{k-1}H_{k-1}P_{k-1}(-) = P_{k-1}(-) - P_{k-1}(+) \quad (2.54)$$

Now pre-multiply both sides of Eq. (2.54) ϕ_{k-1} and post-multiply the same by ϕ_{k-1}^T to obtain:

$$\phi_{k-1}K_{k-1}H_{k-1}P_{k-1}(-)\ \phi_{k-1}^{T} = \phi_{k-1}(P_{k-1}(-) - P_{k-1}(+))\ \phi_{k-1}^{T} \quad (2.55)$$

$$= \phi_{k-1}P_{k-1}(-)\ \phi_{k-1}^{T} - \phi_{k-1}P_{k-1}(+)\ \phi_{k-1}^{T} \quad (2.56)$$

$$= P_k(-) - P_k(+) = K_kH_kP_k(-) \quad (2.57)$$

from Lemma #2 and Eq. (2.19) of the PKF. Therefore, we have shown that

$$\phi_{k-1}K_{k-1}H_{k-1}P_{k-1}(-)\ \phi_{k-1}^{T} = K_kH_kP_k(-) \quad (2.58)$$

Summary

This section demonstrates the parallel Kalman filter, originally developed by Travassos (1985), based upon decoupling the filter's predictor and corrector equations, is optimal in the same sense as the standard Kalman filter. This was proven via two lemmas and two theorems. Since the PKF method can be extended to more than two processors, it is anticipated that the optimality proof can be extended as well. Furthermore, extending the results to nonlinear filtering is also anticipated to be successful.

2.6 GENERALIZATION TO n (EVEN) PROCESSORS

Now that the stability and convergence of the two-processor parallel Kalman filter (PKF) has been analytically investigated and shown to be "mathematically equivalent" to the standard Kalman filter (SKF), we now focus our efforts on extending the PKF algorithm to run on n (even) processors. Our approach is based on the principle of induction; i.e., we develop the two-processor then the four-processor case and then by induction generalize the method for n (even) processors.

To provide a more accurate solution, the generalized method is based on the trapezoidal rule of numerical integration rather than Euler integration which has been used to date. The Kalman filter update is then accurate to $0(h^2)$ rather than $0(h)$. The additional accuracy is important because it is anticipated that the integration step size, h, will be large due to the computational complexity of the Kalman filter equations. For example, with a sample rate of 100 Hz, the Euler-integration-based PKF would be accurate to $0(h) = 0.01$, while the trapezoidal-rule-based PKF would be accurate to $0(h^2) = 0.0001$ (i.e., as accurate as the 12-bit sensor data).

2.6.1 PKF Based on the Trapezoidal Rule (Two Processors)

The trapezoidal rule for integrating a set of ordinary differential equations is given by:

$$x_{k+1} = x_k + \tfrac{1}{2} h(f(x_k,t_k) + f(x_{k+1},t_{k+1})) \tag{2.59}$$

where x is the solution of the ode, $f(x,t)$ is the right-hand-side (RHS) of the initial value problem and $h = t_{k+1} - t_k$ is the integration step size. The trapezoidal rule is an implicit method since x_{k+1} appears implicitly on the RHS of Eq. (2.59). Note that $f(x,t)$ can be, in general, a nonlinear function or linear such as $f(x,t) = Fx(t)$. To solve Eq. (2.59), a predictor is needed of the form below to estimate x_{k+1}.

$$x_{k+1} = x_k + hf(x_k,t_k) \tag{2.60}$$

Hence, combining Eqs. (2.59) and (2.60), we obtain a predictor-corrector method based on the trapezoidal rule:

$$\text{Predictor:} \quad x^p_{k+1} = x^c_k + hf(x^c_k,t_k) \tag{2.61}$$

$$\text{Corrector:} \quad x^c_{k+1} = x^c_k + \tfrac{1}{2} h(f(x^c_k,t_k) + f(x^p_{k+1},t_{k+1})) \tag{2.62}$$

Note that the predictor must be evaluated <u>before</u> the corrector equation can be computed. A parallel predictor-corrector (PPC) method has been derived by Miranker (1967) that allows the predictor and corrector to be evaluated simultaneously on two (2) processors as follows:

<u>Parallel Trapezoidal Rule (Two Processors)</u>:

$$\text{Predictor:} \quad x^p_{k+1} = x^c_{k-1} + 2hf^p_k \tag{2.63}$$

$$\text{Corrector:} \quad x^c_k = x^c_{k-1} + \tfrac{1}{2}h(f^p_k + f^c_{k-1}) \tag{2.64}$$

where $f^p_k = f(x^p_k,k)$ and $f^c_{k-1} = f(x^c_{k-1},k-1)$.

In the special case when $f^p_k = \phi_k x_k(-)$ is the RHS of the Kalman filter state update before a measurement and $G^p_k = \phi_k(I-K_kH_k)P_k(-)(I-K_kH_k)^T\phi_k^T$ is the RHS of the covariance update before a measurement, then the two (2) processor parallel Kalman filter can be derived as follows:

<u>Parallel Kalman Filter Based on Trapezoidal Rule (Two Processors)</u>

$$\text{Predictor:} \quad \hat{x}_{k+1}(-) = \hat{x}_{k-1}(+) + 2h\phi_k\hat{x}_k(-) \tag{2.65}$$

$$P_{k+1}(-) = P_{k-1}(+) + 2hG^p_k \tag{2.66}$$

where $G_k^p = \phi_k(I - K_kH_k)P_k(-) + P_k(-)(I - K_kH_k)^T \phi_k^T$ (2.67)

Corrector: $\hat{x}_k(+) = \hat{x}_{k-1}(+) + h/2(\phi_k\hat{x}_k(-) + \phi_{k-1}\hat{x}_{k-1}(+)) + K_k(z_k - H_k\hat{x}_k(-))$ (2.68)

$P_k(+) = P_{k-1}(+) + h/2(G_k^p + G_{k-1}^c) - K_kH_kP_k(-)$ (2.69)

where $G_k^p = \phi_k(I - K_kH_k)P_k(-) + P_k(-)(I - K_kH_k)^T \phi_k^T$ (2.70)

$G_{k-1}^c = \phi_{k-1}P_{k-1}(+) + P_{k-1}(+) \phi_{k-1}^T$ (2.71)

In the above PKF, the (-) notation represents a value before a measurement update and the (+) notation is a value after a measurement update. Similarly, the p for predictor corresponds to the (-) notation and the c for corrector value corresponds to the (+) notation.

2.6.2 PKF Based on the Trapezoidal Rule (Four Processors)

With four (4) processors, the parallel trapezoidal rule is given by:

Parallel Trapezoidal Rule (Four Processors:

Predictor: $x_{2k+2}^p = x_{2k-2}^c + 4hf_{2k}^p$ (2.72)

$x_{2k+1}^p = x_{2k-2}^c + 3/2\ h(f_{2k}^p + f_{2k-1}^p)$ (2.73)

Corrector: $x_{2k}^c = x_{2k-3}^c - h/2\ (3f_{2k}^p - 9f_{2k-1}^p)$ (2.74)

$x_{2k-1}^c = x_{2k-3}^c + 2hf_{2k-2}^c$ (2.75)

where $f_{2k}^p = f(x_{2k}^p, 2k)$, $f_{2k-1}^p = f(x_{2k-1}^p, 2k-1)$

and $f_{2k-2}^c = f(x_{2k-2}^c, 2k-2)$

In the case where

$f_{2k}^p = \phi_{2k}\ \hat{x}_{2k}(-)$, (2.76)

$f_{2k-1}^p = \phi_{2k-1}\ \hat{x}_{2k-1}(-)$ and (2.77)

$f_{2k-2}^c = \phi_{2k-2}\ \hat{x}_{2k-2}(+)$ (2.78)

are the RHS of the state update in the Kalman filter and

$G_{2k}^p = \phi_{2k}(I - K_{2k}H_{2k})\ P_{2k}(-) + P_{2k}(-)\ (I - K_{2k}H_{2k})^T\ \phi_{2k}^T$ (2.79)

$G_{2k-1}^p = \phi_{2k-1}(I - K_{2k-1}H_{2k-1})P_{2k-1}(-) + P_{2k-1}(-)(I - K_{2k-1}H_{2k-1})^T\phi_{2k-1}^T$ (2.80)

and $G_{2k-2}^c = \phi_{2k-2}\ P_{2k-2}(+) + P_{2k-2}(+)\ \phi_{2k-2}^T$ (2.81)

Parallel Kalman Filter Based on Trapezoidal Rule (Four Processors):

Predictor:
$$\hat{x}_{2k+2}(-) = \hat{x}_{2k-2}(+) + 4h\,\phi_{2k}\hat{x}_{2k}(-) \tag{2.82}$$
$$\hat{x}_{2k+1}(-) = \hat{x}_{2k-2}(+) + 3h/2\,(\phi_{2k}\hat{x}_{2k}(-) + \phi_{2k-1}\hat{x}_{2k-1}(-)) \tag{2.83}$$
$$P_{2k+2}(-) = P_{2k-2}(+) + 4hG^{p}_{2k} \tag{2.84}$$
$$P_{2k+1}(-) = P_{2k-2}(+) + 3h/2\,(G^{p}_{2k} + G^{p}_{2k-1}) \tag{2.85}$$
where
$$G^{p}_{2k} = \phi_{2k}(I - K_{2k}H_{2k})\,P_{2k}(-) + P_{2k}(-)(I-K_{2k}H_{2k})^{T}\phi^{T}_{2k} \tag{2.86}$$
$$G^{p}_{2k-1} = \phi_{2k-1}(I - K_{2k-1}H_{2k-1})P_{2k-1}(-) + P_{2k-1}(-)(I - K_{2k-1})^{T}\,\phi^{T}_{2k-1} \tag{2.87}$$

Corrector:
$$\hat{x}_{2k}(+) = \hat{x}_{2k-3}(+) - h/2\,(3\phi_{2k}\hat{x}_{2k}(-) - 9\,\phi_{2k-1}\hat{x}_{2k-1}(-)) + K_{2k}(z_{2k} - H_{2k}\hat{x}_{2k}(-)) \tag{2.88}$$
$$\hat{x}_{2k-1}(+) = \hat{x}_{2k-3}(+) + 2h\,\phi_{2k-2}\hat{x}_{2k-2}(+) + K_{2k-1}(z_{2k-1} - H_{2k-1}(-)\,\hat{x}_{2k-1}(-)) \tag{2.89}$$
$$P_{2k}(+) = P_{2k-3}(+) - h/2\,(3G^{p}_{2k} - 9G^{p}_{2k-1}) + (I - K_{2k}H_{2k})P_{2k}(-) \tag{2.90}$$
$$P_{2k-1}(+) = P_{2k-3}(+) + 2hG^{c}_{2k-2} + (I - K_{2k-1}H_{2k-1})P_{2k-1}(-) \tag{2.91}$$
where G^{p}_{2k} and G^{p}_{2k-1} are defined above and
$$G^{p}_{2k-1} = \phi_{2k-1}P_{2k-1}(+) + P_{2k-1}(+)\,\phi^{T}_{2k-1} \tag{2.92}$$
$$G^{c}_{2k-2} = \phi_{2k-2}P_{2k-2}(+) + P_{2k-2}(+)\,\phi^{T}_{2k-2} \tag{2.93}$$

2.6.3 PKF Based on Trapezoidal Rule (n (even) Processors

In the previous sections, the PKF equations were derived for the two processor and four processor cases. In general, $n = 2^{s}$ (even) processors are needed to implement the PKF equations. If we let $m = 2^{s-1}$ and $q = m-1$, the PKF equations can be generalized to run on n processors. The RHS definitions for the generalized PKF equations are defined as follows.

$$f^{p}_{mk} = \phi_{mk}\,\hat{x}_{mk}(-) \tag{2.94}$$
$$f^{p}_{mk-1} = \phi_{mk-1}\,\hat{x}_{mk-1}(-) \tag{2.95}$$
.
.
.

$$f^{p}_{mk-m} = \phi_{mk-m}\ \hat{x}_{mk-m}(-) \tag{2.96}$$

$$f^{c}_{mk-m} = \phi_{mk-m}\ \hat{x}_{mk-m}(+) \tag{2.97}$$

With these definitions, the PKF for n (even) processors are summarized in Eqs. (2.98) to (2.115) on the following pages.

2.7 SUMMARY

Although it was shown that the PKF equations can be generalized for n (even) processors, the following recommendation is made based on our experience with parallel computing. It is recommended that the four processor PKF equations be used to propagate each state (or covariance) equation separately in a nine-state target tracking application. With four processors per state, a 4x speed-up is possible per state equation update. With four processors on a card, the nine-state filter could be implemented with nine cards or 9x4 = 36 processing elements. This approach may be computationally easier to manage and provide a 9x4 = 36x speed-up compared with partitioning the original PKF equations to run on 36 processors. The main concern is that when the large number of algebraic terms that arise from the decoupling are computed, human error in the algebraic partitioning could lead to inaccurate tracking. The partitioning can be automated by a computer to minimize this potential problem. In addition, it is well known that beyond say 32 processors, the benefit of parallel processing is less than linear and adding more processors may not speed up overall computations that much. Hence, using four processors per state (or covariance) equation provides maximum benefit, ease of partitioning and reduces the potential for human error in the decoupling.

Parallel Kalman Filter Based on Trapezoidal Rule

Generalization to n (even) Processors

State Predictor Equations:

$$\hat{x}_{mk+m}(-) = \hat{x}_{mk-m}(+) + nhf^{p}_{mk} \tag{2.98}$$

$$\hat{x}_{mk+m-1}(-) = \hat{x}_{mk-m}(+) + (n-1)/2h(f^{p}_{mk} + f^{p}_{mk-1}) \tag{2.99}$$

$$\hat{x}_{mk+m-2}(-) = \hat{x}_{mk-m}(+) + (n-2)/2h(f^{p}_{mk} + f^{p}_{mk-1} + f^{p}_{mk-2}) \tag{2.100}$$

$$\hat{x}_{mk+m-3}(-) = \hat{x}_{mk-m}(+) + (n-3)/2h(f^{p}_{mk} + f^{p}_{mk-1} + f^{p}_{mk-2} + \ldots + f^{p}_{mk-m}) \tag{2.101}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$\hat{x}_{mk+m-q}(-) = \hat{x}_{mk-m}(+) + (n-q)/2h(f^{p}_{mk} + f^{p}_{mk-1} + f^{p}_{mk-2} + \ldots + f^{p}_{mk-m}) \tag{2.102}$$

State Corrector Equations:

$$\hat{x}_{mk}(+) = \hat{x}_{mk-n+1}(+) + (n-q)/2h(f^{p}_{mk-m+q} + \ldots + f^{p}_{mk-m+2} + f^{p}_{mk-m+1} + f^{p}_{mk-m}) \tag{2.103}$$

$$\hat{x}_{mk-1}(+) = \hat{x}_{mk-n+1}(+) + (n-2)/2h(f^{p}_{mk-m+2} + f^{p}_{mk-m+1} + f^{p}_{mk-m}) \tag{2.104}$$

$$\hat{x}_{mk-2}(+) = \hat{x}_{mk-n+1}(+) + (n-1)/2h(f^{p}_{mk-m+1} + f^{p}_{mk-m}) \tag{2.105}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$\hat{x}_{mk-q}(+) = \hat{x}_{mk-n+1}(+) + n/2h(3^{q}f^{p}_{mk} + q3^{3q}f^{p}_{mk-m} + f^{c}_{mk-m}) \tag{2.106}$$

Definitions

For the parallel predictor-corrector generalization, define the following variables:

n = the number of parallel processing elements = 2^{s} (i.e., a power of 2)

$m = 2^{s-1}$ = the number of processors used by the predictor (or corrector)

$q = m-1$ = a convenient definition

Parallel Kalman Filter Based on Trapezoidal Rule

Generalization to n (even) Processors

Covariance Predictor Equations:

$$P_{mk+m}(-) = P_{mk-m}(+) + nhG^P_{mk} \tag{2.107}$$

$$P_{mk+m-1}(-) = P_{mk-m}(+) + (n-1)/2h\ (G^P_{mk} + G^P_{mk-1}) \tag{2.108}$$

$$P_{mk+m-2}(-) = P_{mk-m}(+) + (n-2)/2h\ (G^P_{mk} + G^P_{mk-1} + G^P_{mk-2}) \tag{2.109}$$

$$P_{mk+m-3}(-) = P_{mk-m}(+) + (n-3)/2h\ (G^P_{mk} + G^P_{mk-1} + G^P_{mk-2} + \ldots + G^P_{mk-m}) \tag{2.110}$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$P_{mk+m-q}(-) = P_{mk-m}(+) + (n-q)/2h\ (G^P_{mk} + G^P_{mk-1} + G^P_{mk-2} + \ldots + G^P_{mk-m}) \tag{2.111}$$

Covariance Corrector Equations:

$$P_{mk}(+) = P_{mk-n+1}(+) + (n-q)/2h\ (G^P_{mk-m+q} + \ldots + G^P_{mk-m+2} + G^P_{mk-m+1} + G^P_{mk-m}) \tag{2.112}$$

$$P_{mk-1}(+) = P_{mk-n+1}(+) + (n-2)/2h\ (G^P_{mk-m+2} + G^P_{mk-m+1} + G^P_{mk-m}) \tag{2.113}$$

$$P_{mk-2}(+) = P_{mk-n+1}(+) + (n-1)/2h\ (G^P_{mk-m+1} + G^P_{mk-m}) \tag{2.114}$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$P_{mk-q}(+) = P_{mk-n+1}(+) + n/2h\ (3^q G^P_{mk} + q3^{3q} G^P_{mk-m} + G^c_{mk-m}) \tag{2.115}$$

SECTION 3

NONLINEAR EXTENDED PARALLEL KALMAN FILTER ALGORITHMS AND ARCHITECTURES BASED ON THE DECOUPLING PRINCIPLE

3.1 BACKGROUND

The SDI sensor processing problem is nonlinear due to coordinate transformations and angle-only measurements. The SDI target math model can be summarized as follows for the extended Kalman filter:

$$\dot{x}(t) = f(x(t),t) + w(t) \qquad 0 \leq t \leq T$$

$$z_k = h(x(t_k)) + v_k \qquad k = 1, 2, 3, \ldots, N$$

The well-known extended Kalman filter for a sequential computer is given by (Gelb, 1974):

Predictor:

$$\dot{\hat{x}}(t) = f(\hat{x}(t)) \qquad \hat{x}(t_0) = x_0$$

$$\dot{P}(t) = F(\hat{x}(t))\, P(t) + P(t)\, F^T(\hat{x}(t)) + Q(t) , \quad P(t_0) = P_0$$

Corrector:

$$\hat{x}_k(+) = \hat{x}(-) + K_k(z_k - h_k(\hat{x}_k(-)))$$

$$P_k(+) = (I - K_k H_k(\hat{x}_k(-)))\, P_k(-)$$

$$K_k = P_k(-)\, H_k^T(\hat{x}_k(-)) * (H_k(\hat{x}_k(-))\, P_k(-) + R_k)^{-1}$$

where

$$F(\hat{x}(t)) = \left.\frac{\partial f(x(t))}{\partial x(t)}\right|_{x(t) = \hat{x}(t)}$$

$$H_k(x(-)) = \left.\frac{\partial h(x(t_k))}{\partial x(t_k)}\right|_{x(t_k) = \hat{x}(-)}$$

3.2 DECOUPLING THE EXTENDED KALMAN FILTER STATE UPDATE

With two processors, the state predictor and corrector can be decoupled by forcing the corrector to lag the predictor by one time step. If a parallel version of the Trapezoidal Rule (Miranker, 1967) is used to integrate the

nonlinear state model equations, the extended Kalman filter state predictor and corrector can be computed as follows with accuracy of $0(h^2)$.

Predictor:

$$\hat{x}_{k+1}(-) = \hat{x}_{k-1}(+) + 2hf_k^p$$

where h is the integration step size and

$$f_k^p = f(\hat{x}_k(-), t_k)$$

Corrector:

$$\hat{x}_k(+) = \hat{x}_{k-1}(+) + h/2(f_k^p + f_{k-1}^c) + K_k(z_k - h(\hat{x}_k(-)))$$

where $f_{k-1}^c = f(\hat{x}_{k-1}(+), t_{k-1})$

With two (2) processors the state predictor and corrector can be computed using a 4th order Parallel Adams-Moulton Method as follows with $0(h^4)$ accuracy.

Predictor:

$$\hat{x}_{k+1}(-) = \hat{x}_{k-1}(+) + h/3(8f_k^p - 5f_{k-1}^c + 4f_{k-2}^c - f_{k-3}^c)$$

where $f_k^p = f(\hat{x}_k(-), t_k)$, $f_{k-2}^c = f(\hat{x}_{k-2}(+), t_{k-2})$ and

$$f_{k-3}^c = f(\hat{x}_{k-3}(+), t_{k-3})$$

Corrector:

$$\hat{x}_k(+) = \hat{x}_{k-1}(+) + h/24(9f_k^p + 19f_{k-1}^c - 5f_{k-2}^c + f_{k-3}^c) + K_k(z_k - h(\hat{x}_k(-)))$$

The key here is that all the values on the right hand side (RHS) of the above predictor/corrector equations are available for simultaneous computation in parallel (see Figure 3-1 below).

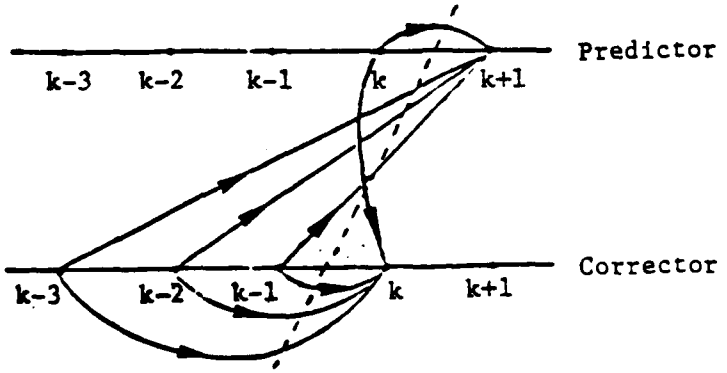


FIGURE 3-1 Computational Wavefront of Extended PKF State Update

3.3 DECOUPLING THE EXTENDED KALMAN FILTER COVARIANCE UPDATE

Let the covariance of the estimation error before and after an update be denoted as in Section 2. Now if we define the RHS of the continuous covariance update as follows, we can compute the extended Parallel Kalman Filter (PKF) covariance in parallel:

$$\dot{P}(t) = F(\hat{x}(t))P(t) + P(t)F^T(\hat{x}(t)) + Q(t) \qquad P(t_0) = P_0$$

$$= G(F(\hat{x}(t)),P(t),Q(t),t)$$

$$\text{where } F(\hat{x}(t)) = \frac{\partial f(\hat{x}(t))}{\partial x(t)} \Bigg|_{x(t) = \hat{x}(t)}$$

Note that G is a matrix differential equation that is nonlinearly related to P.

With two (2) processors using the parallel trapezoidal rule, the extended Kalman filter covariance update can be computed with $0(h^2)$ accuracy as follows:

$$\text{Predictor: } P_{k+1}(-) = P_{k-1}(+) + 2hG_k^p$$

$$\text{where } G_k^p = G(F(\hat{x}_k(-)),P_k(-),Q_k(-),t_k)$$

$$\text{Corrector: } P_k(+) = P_{k-1}(+) + h/2(G_k^p + G_{k-1}^c) + (I - K_kH_k(\hat{x}_k(-))P_k(-)$$

$$\text{where } G_{k-1}^c = G(F(\hat{x}_{k-1}(+)),\ P_{k-1}(+),Q_{k-1}(+),t_{k-1})$$

$$\text{and } H_k(\hat{x}_k(-)) = \frac{\partial h(x(t_k))}{\partial x(t_k)} \Bigg|_{x(t_k) = \hat{x}_k(-)}$$

With two (2) processors the covariance predictor and corrector equations can be computed using the 4th order Parallel Adams Moulton Method with $0(h^4)$ accuracy as follows:

$$\text{Predictor: } P_{k+1}(-) = P_{k-1}(+) + h/3(8G_k^p - 5G_{k-1}^c + 4G_{k-2}^c - G_{k-3}^c)$$

$$\text{where } G_{k-1}^c = G(F(\hat{x}_{k-1}(+)),P_{k-1}(+),Q_{k-1}(+),t_{k-1})$$

$$G_{k-2}^c = G(F(\hat{x}_{k-2}(+)),P_{k-2}(+),Q_{k-2}(+),t_{k-2})$$

$$G_{k-3}^c = G(F(\hat{x}_{k-3}(+)),P_{k-3}(+),Q_{k-3}(+),t_{k-3})$$

$$\text{Corrector: } P_k(+) = P_{k-1}(+) + h/24(9G_k^p + 19G_{k-1}^c - 5G_{k-2}^c + G_{k-3}^c)$$

$$- K_kH_k(\hat{x}_k(-))P_k(-)$$

In both the two and four processor extended Parallel Kalman Filter

equations, the Kalman gain is computed as follows:

$$\text{Kalman Gain: } P_k(-)H^T(\hat{x}_k(-)*(H_k(\hat{x}_k(-))P_k(-)H_k(\hat{x}_k(-)) + R_k)^{-1}$$

The extended parallel Kalman filter algorithms presented in this section can be generalized to run on a large number of processors. The four processor extended PKF based on the parallel Adams Moulton method is of major interest because the 4th order parallel Adams Moulton method is fast (because the equations can be computed on four processors four times faster than on a sequential computer) and accurate to $0(h^4)$ to give eight digit or more precision. In addition, this method is well matched for execution on a parallel computer with four processors per node (like the Systolic-481 board discussed in Section 4. A candidate architecture for evaluating the 4th order parallel Adams Moulton method is given in Figure 3-2 below.

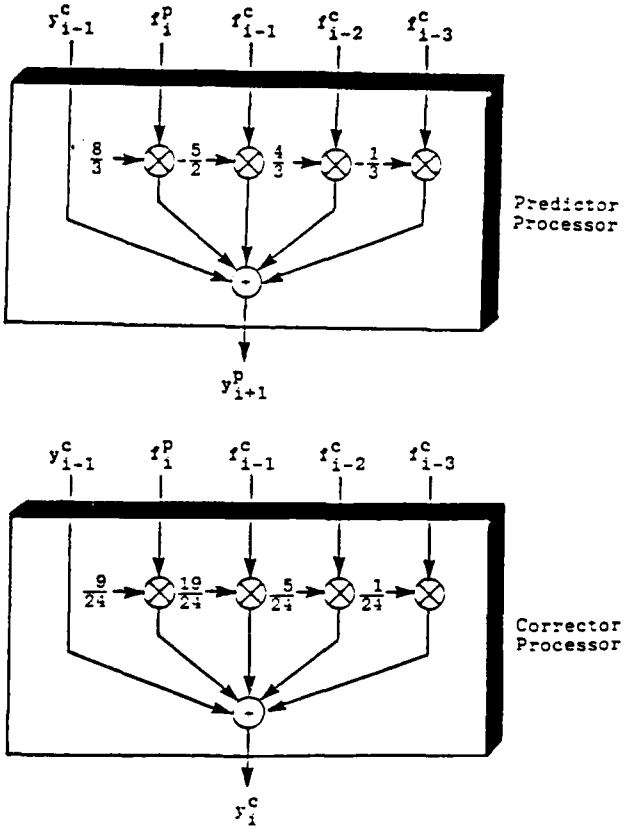


FIGURE 3-2 Parallel Architecture for the 4th Order Adams Moulton Method

SECTION 4

WORDLENGTH, MEMORY AND PARALLEL PROCESSOR SELECTION

Before implementing the parallel Kalman filter algorithms and architectures derived previously in this report, it is of interest to examine the wordlength, memory and timing requirements for these methods. The timing requirements, in particular, have a direct impact on the selection of the VLSI arithmetic processors used for computation. Memory speed and bus considerations also impact overall system throughput. To start, wordlength considerations are analyzed first.

4.1 ERRORS IN THE KALMAN GAIN

Suppose that due to numerical inaccuracies, the actual Kalman gain consists of two parts:

$$\underset{\text{actual Kalman gain}}{K} + \underset{\text{exact Kalman gain}}{K} + \underset{\text{error in Kalman gain}}{\Delta K}$$

Then the finite wordlength effects of the mantisa in floating point arithmetic on the Kalman filter can be summarized by the following theorem:

Theorem 4.1: For the linear Kalman filter in Section 2, given that inaccuracies exist in the Kalman gain (i.e., $K = K + \Delta K$), the number of bits needed in the mantisa to ensure stability is given by:

$$b \geq -\log_2\left(\frac{2\epsilon}{n}\,\frac{|H|}{|HP(-)H^T|}\right)$$

where

$\epsilon > 0$ is the error tolerance, n is the system order and

$$|\Delta K| \leq \frac{n\,2^{-b}}{2} \quad .$$

Corollary: For $H = I$ and $P(-) = I$, the number of bits needed in the mantissa to ensure stability can be estimated as follows:

$$b \geq -\log_2\left(\frac{2\epsilon}{n}\right)$$

Proof: Substituting $K = K + \Delta K$ into the $P(+)$ update we have:

$$P(+) = (I - (K + \Delta K)H)\, P(-) \geq \epsilon \tag{4.2}$$

$$= \underbrace{(I - KH)\, P(-)}_{P(+)} - \Delta K H P(-) \geq \epsilon \tag{4.3}$$

$$\rightarrow \quad -\Delta K H P(-) \geq \epsilon \tag{4.4}$$

Now multiplying both sides of Eq. (4.3) by $-H^T$ gives:

$$\Delta K \underbrace{HP(-)H^T}_{\text{scalar} > 0 \text{ if } P(-) > 0} \leq -\epsilon H^T \tag{4.5}$$

Since $HP(-)H^T > 0$ and a scalar, we can divide both sides of Eq. (4.4) by $HP(-)H^T$. Hence, we have:

$$\Delta K \leq -\frac{\epsilon H^T}{HP(-)H^T} \tag{4.6}$$

since the $|H^T| = |H|$. Now suppose $|\Delta K| \geq \frac{n}{2}\, 2^{-b}$, then

$$\frac{n}{2}\, 2^{-b} \leq |\Delta K| \leq \frac{\epsilon |H|}{|HP(-)H^T|} \tag{4.8}$$

Taking the $\log_2$ of both sides gives:

$$-b \leq \log_2\left(\frac{2\epsilon}{n}\, \frac{|H|}{|HP(-)H^T|}\right) \tag{4.9}$$

or equivalently

$$b \geq -\log_2\left(\frac{2\epsilon}{n}\, \frac{|H|}{|HP(-)H^T|}\right) \tag{4.10}$$

Note that if $H = I$ and $P(-) = I$, the desired result is obtained as follows:

$$b \geq -\log_2\left(\frac{2\epsilon}{n}\right) \tag{4.11}$$

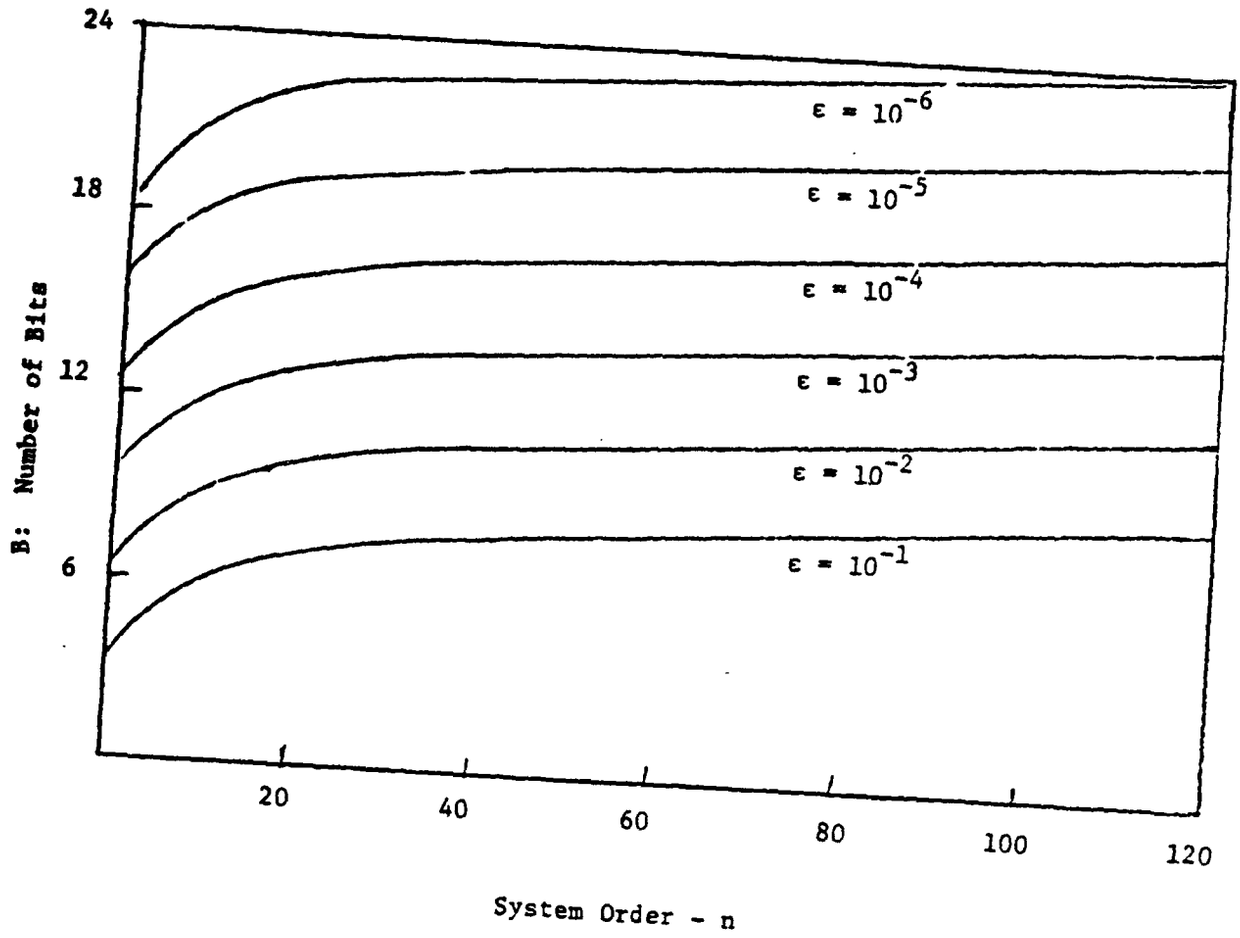


FIGURE 4-1 Mantissa Wordlength Requirement for Stable Kalman Filtering

The result of Theorem 4.1 is particularly interesting when plotted for various values of ε and n (see Figure 4-1). Note that as the system order (n) increases so does the mantissa wordlength requirements. Also note that scalar systems require a minimum precision of bits for one decimal digit of accuracy. Six digit accuracy requires at least 24-bits in the mantissa. More than six digit accuracy requires 32-bit floating point.

4.2 ERRORS IN THE COVARIANCE UPDATE

Another approach in analyzing the wordlength requirements in the Kalman filter is to study errors in the covariance update. As it turns out, a companion result can be derived similar to Theorem 4.1. The companion result is summarized in the following theorem:

Theorem 4.2: For the Kalman filter in Section 2 given that inaccuracies exist in the covariance update (i.e., $P(-) = P(-) + \Delta P(-)$), the number of bits needed in the mantissa to ensure stability is given by:

$$b \geq -\log_2 \left(\frac{2\varepsilon}{n} \left| (I - KH)^{-1} \right| \right)$$

where

$\varepsilon > 0$ is the error tolerance, n is the system order and

$$|\Delta P| \leq \frac{n}{2} 2^{-b} .$$

Corollary: For $KH \to 0$ (i.e., in the steady state) the number of bits needed in the mantissa to ensure stability can be estimated as follows:

$$b \geq -\log_2 \left(\frac{2\varepsilon}{n} \right)$$

Proof: Suppose $P(-) = P(-) + \Delta P(-)$.

($P(-)$: actual covariance; $P(-)$: exact covariance; $\Delta P(-)$: error in covariance)

Then
$$P(+) = \underbrace{(I - KH)P(-)}_{P(+)} + (I - KH)\,\Delta P(-) \geq \varepsilon \tag{4.12}$$

$$\rightarrow (I - KH)\,\Delta P(-) \geq \varepsilon \tag{4.13}$$

Now multiplying both sides of Eq. (4.13) by $(I - KH)^{-1}$ we have:

$$\Delta P(-) \geq \varepsilon (I - KH)^{-1} \tag{4.14}$$

Taking the norm of both sides of Eq. (4.14) and using the fact that $|\Delta P(-)| \leq \frac{n}{2} 2^{-b}$ results in:

$$\frac{n}{2} 2^{-b} \leq \varepsilon \left| (I - KH)^{-1} \right| \tag{4.15}$$

Taking the $\log_2$ of both sides of Eq. (4.15) gives:

$$-b \leq \log_2 \left(\frac{2\varepsilon}{n} \left| (I - KH)^{-1} \right| \right) \tag{4.16}$$

Multiply both sides of Eq. (4.16) by -1 giving the desired result:

$$b \geq -\log_2\left(\frac{2\varepsilon}{n}\left|(I - KH)^{-1}\right|\right) \tag{4.17}$$

Note that if $KH \to 0$ in Eq. (4.17) we have:

$$b \geq \log_2\left(\frac{2\varepsilon}{n}\right) \tag{4.18}$$

Note that the results of Theorems 4.1 and 4.2 are very similar and provide the same results under the conditions that $H = I$, $P(-) = I$ and $KH \to 0$.

4.3 MEMORY CONSIDERATIONS

The parallel Kalman filter algorithms and architectures derived in Sections 2 and 3 use decoupling to permit the predictor and corrector equations to be computed on separate processors. At any given time step k, the state, covariance, and measurements must be stored, as well as intermediate values associated with the linear PKF's matrix/vector calculations. For a typical nine state filter, the operation count given in Section 1 indicates that the number of operations in the standard Kalman filter is

additions:	$n \times n + 2n - 1 = 98$
multiplications:	$2n \times n + 4n + 1 = 199$
divisions:	1

when $n = 9$. The data storage requirement is on the order of 8 bytes x (98 + 199 - 1) = 2384 bytes or 2K bytes for 64-bit precision. If the linear two processor PKF developed in Section 2 is used, it must be initialized by running the standard SKF for the first two (2) time steps. Then the dual (2) processor PKF can be run at step 3 (i.e., at $k = 3$). Hence, the following values of x, ϕ, P, z, K, H and R must be stored in memory.

State Vector Values:	$\hat{x}_0(+)$, $\hat{x}_1(+)$, $\hat{x}_0(-)$, $\hat{x}_1(-)$, $\hat{x}_2(-)$
State Transition Matrix Values:	ϕ_0, ϕ_1, ϕ_2
Covariance Matrix Values:	$P_0(+)$, $P_1(+)$ $P_1(-)$, $P_2(-)$
Kalman Gain Values:	K_1, K_2
Others:	H_1, H_2, R_1, R_2

Once the linear two processor PKF is initialized, memory is needed to store the updated values of x, ϕ, P, z, H and R. Hence, in general, the overall memory requirement is:

Memory = (np + 1) x (storage requirement of the standard Kalman filter)

where np = the number of parallel processing elements.

Thus, (np + 1) x (2 Kbytes) are needed to store the data in the parallel filter. With np = 2, this corresponds to 6 Kbytes. With np = 32, this corresponds to approximately 64K bytes of RAM.

Note that the above memory sizing is for data only. The PKF program memory has not been sized. Because the two processor PKF algorithm can be coded with less than 1000 lines of code and the compiled version of a 1000 line program requires about 128 Kbytes of RAM to store, a reasonable estimate of the storage requirement for the linear PKF program would be:

PKF Program memory = np x 128 Kbytes

where np = the number of parallel processing elements.

With 32 processors, the program memory is, therefore, estimated to be 32 x 128 Kbytes = 4 Mbytes.

Hence, a parallel processor with 4 Mbytes of bulk memory (i.e., relatively slow DRAM) and 64 Kbytes of fast RAM (i.e., cache memory) should be capable of implementing a 32 processor linear PKF.

Because nonlinear function evaluation generally results in more intermediate values than linear matrix/vector operations, the amount of local data storage might be increased by a factor of four. Hence, 4 x 64K = 256K of fast RAM is recommended for nonlinear extended parallel Kalman filter data storage. Four Mbytes of program memory should be sufficient, however, for the nonlinear PKF.

4.4 PARALLEL PROCESSOR SELECTION

One method of estimating the computational requirements for the parallel Kalman filter is to total the number of additions, multiplications and divisions needed to complete one cycle of the Kalman filter algorithm. For example, the simple Kalman filter algorithm defined in Section 1 requires only 98 additions, 199 multiplications and 1 division per cycle for a nine-state filter. Hence, at 100 cycles per second (i.e., 100 Hz sample rate) the number of arithmetic operations is given by 100 x 298 = 29,800 operations per second. Ideally, a microprocessor capable of 33.1 μsec per operation is all that is needed to implement a nine-state filter. Hence, a single Motorola 68020/68881

pair can easily handle the computational requirements of the Kalman filter assuming 100% efficiency. Note that 33.6 μsec per pass through the filter corresponds to an update rate of 2,800 samples per second.

Typically, however, only 10 to 30% of peak performance is achieved in practice due to data bus and memory access time restrictions. Therefore, one target may be updated at a 4000 updates per second rate. 100 targets may be updated at a 4 Hz rate. 10,000 targets at a 0.4 Hz rate (every 2.5 seconds). For nonlinear filtering, typical of SDI target tracking problems, 64-bit precision and the need to compute trigonometric functions for coordinate transformations can slow computations down by one or perhaps two orders of magnitude (10x to 100x).

Since it is well known that Kalman filtering must be performed using floating-point arithmetic to avoid stability problems, the only viable method to gain back the throughput for nonlinear SDI filtering problems using an extended Kalman filter is with parallel processing. Optical processing is fast but optical fixed point can cause stability problems with the Kalman filter. Therefore, to rapidly implement the parallel Kalman filter with 32/64-bit floating-point precision, an aggregate computation rate of 10,000 x 29,800 = 298 million operations per second is needed to track 10,000 targets simultaneously. A parallel processing system of 16 x 16 = 256 processors needs a computation rate of 1.17 MFLOPs per processor to perform the necessary computations. Using four (4) 25 MHz Motorola 68881 math coprocessors per board, 1.26 MFLOP performance is readily achievable. Hence, with 256 boards it it feasible to track 10,000 targets in real time.

The general-purpose nature of the Motorola 68020/68881 processors is well suited for nonlinear, as well as linear, Kalman filtering. In particular because trigonometric functions (sin, cos, tag, etc.) and square roots commonly occur in coordinate transformations associated with SDI sensor processing, high-speed general-purpose hardware (such as the Systolic-481 parallel numeric processor) is needed to handle the throughput requirements (see Figure 4-2).

The Systolic-481 is a Vme bus compatible parallel numeric processor board capable of full IEEE-P754 standard 32-bit, 64-bit and 80-bit floating-point computations. The Systolic-481 contains one (1) Motorola 68020, four (4) 68881 numeric coprocessors and 256 Kbytes of 70 nsec static RAM on a single

233.35 mm x 160 mm board. Multiple 481 boards can be installed into a Vme bus system for additional performance gains approaching a CRAY supercomputer. The advantage of the Systolic-481, however, is that it is very compact. A scientific software library of more than 400 commonly-used subroutines (over 200 separate functions) is available for the Systolic-481 simplifying software development. Because the Systolic-481 rapidly performs linear, as well as nonlinear, transcendental and trigonometric functions, it is well matched for SDI battle management computations.

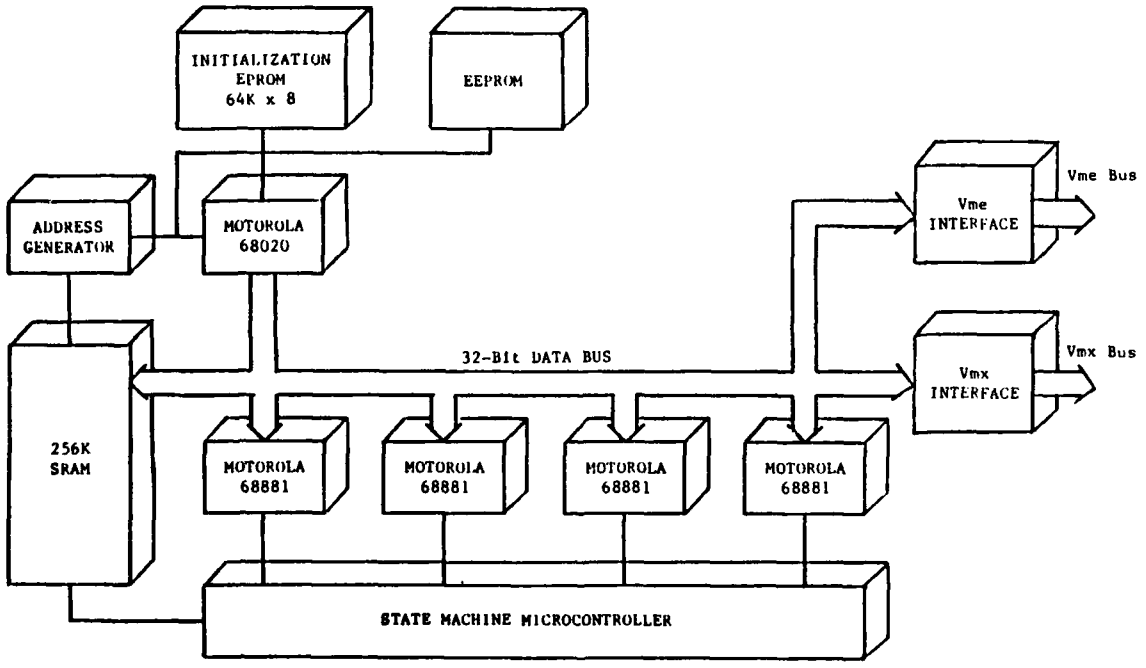


FIGURE 4-2 Systolic-481 Vme Bus Architecture

Table 4-1: Systolic-481 Hardware Specifications (One Board)*

Double Precision Floating-Point Hardware:
One Motorola 68020 and four Motorola 68881 math coprocessors on board
Processor Speeds: 16.67 MHz, 20.0 MHz and 25.0 MHz

Floating-Point Data Format:
IEEE-P754 standard
32-bit single precision, 8-bit signed exponent, 23-bit mantissa
64-bit single precision, 11-bit signed exponent, 52-bit mantissa
80-bit single precision, 15-bit signed exponent, 64-bit mantissa

Integer Arithmetic Formats:
8-bit byte integer
16-bit word integer
32-bit long integer

Data Transfer:
Data transfers conform to Vme bus protocol and performance. External DMA transfer at 120 nsec per cycle results in a 66.7 Mbyte per second transfer rate.

On Board Memory:
256 Kbytes, 64K x 32-bit, 70 ns static RAM

Data Registers:
Four stacks of 8 80-bit data registers (32 total)

Stack Pointers:
Two 32-bit pointers

Constants:
22 on-chip ROM constants

Support Functions:
Full set of trigonometric and transcendental functions

Bus Interface:
VME, 24-bit address (16 Mbytes), 16/32-bit data transfer

High-Speed Auxiliary Port:
32-bit bus on Vmx port

Board Size:
233.35mm x 160.00mm (optional extension bracket to 233.35mm to 220mm)

Power Requirements:
+5V, 5amps

* Multiple 481 boards can be operated in parallel if additional computational power is needed.

Table 4-2: Systolic-481 Double Precision Benchmarks (One Board)

64-bit Floating Point

Addition	1.08 usec
Subtraction	1.08 usec
Multiplication	1.36 usec
Division	1.81 usec
Square Root	1.84 usec
Sine	6.62 usec
Cosine	6.62 usec
Tangent	6.93 usec
Exponential	7.26 usec
Logorithm	7.65 usec

Note:

1.) Assumes 16.667 MHz clock and 4 Motorola 68881 math coprocessors are operating in parallel at peak performance. With a 25 MHz clock, computation time is reduced by 33%. For example, the 64-bit sine time would be only 4.4 usec running at 25 MHz.

2.) The proposed testbed includes 8 Systolic-481 boards which can operate simultaneously in parallel. With 8 boards the effective computation time of the system is 1/8 of the above. For example, with a 25 MHz clock, a 64-bit sine value could be computed in 4.4 usec/8 = 0.55 usec or 550 nsec. This assumes, of course, that at least 4 sine values need to be computed as is the case in coordinate transformations during SDI measurement processing. Thus, the 32-processor testbed compares favorably with a CRAY supercomputer.

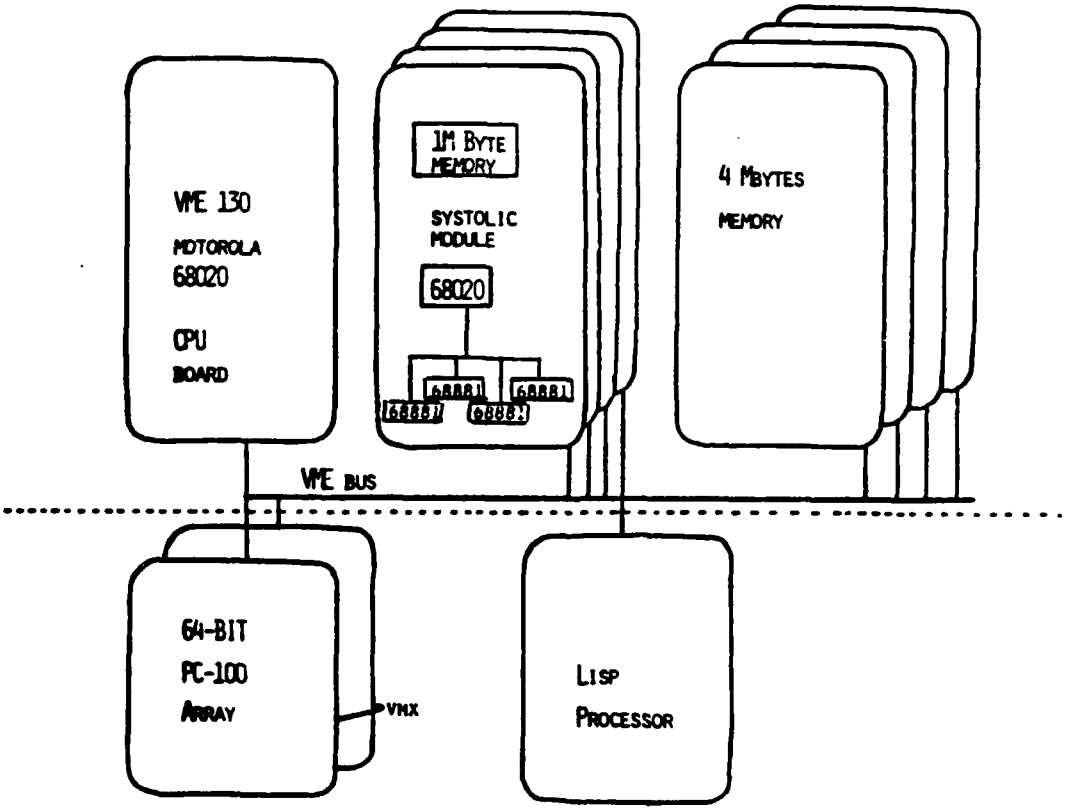


Figure 4-3: Systolic-481 Parallel Architecture

SECTION 5

APPLICATION OF THE PKF TO SDI SENSOR TRACK PROCESSING

The SDI sensor processing problem can be partitioned into a set of simpler tasks which when "chained" together provide information to the battle manager. Figure 5-1 illustrates the major parts of the sensor data processing required by the SDI program. Although each major block in Figure 5-1 can benefit from high-speed computation, this report is concerned with scan-to-scan correlation and track processing since Kalman filtering is generally required for precision tracking. Thus, the remainder of this section formulates the SDI track processing problem as it applies to the parallel Kalman filtering algorithms and architectures developed under our Phase I SBIR effort.

5.1 BACKGROUND

To show the effectiveness/payoff of the proposed research it is important to consider a meaningful SDI problem. The SDI problem should be representative of typical ballistic missile applications and serve as a baseline to measure the benefits/accuracy of the Phase I SBIR parallel Kalman filter technology. With this in mind, the following candidate test problem is recommended. Although this test problem is relatively simple, it illustrates the computations which arise in SDI sensor track processing.

A Simple Test Problem

Consider the problem of estimating the position of an object (missile) from angle-only (or range) measurements. The geometry of this two-dimensional tracking problem is illustrated in Figure 5-2. Typical parameter values for this exercise are given in Table 5-1.

In Figure 5.1, X_t, Y_t represents the target position in X-Y coordinates and Y_s represents the sensor position.

The target range is given by

$$Z_t = (X_t^2 + (Y_t - Y_s)^2)^{\frac{1}{2}} \tag{5.1}$$

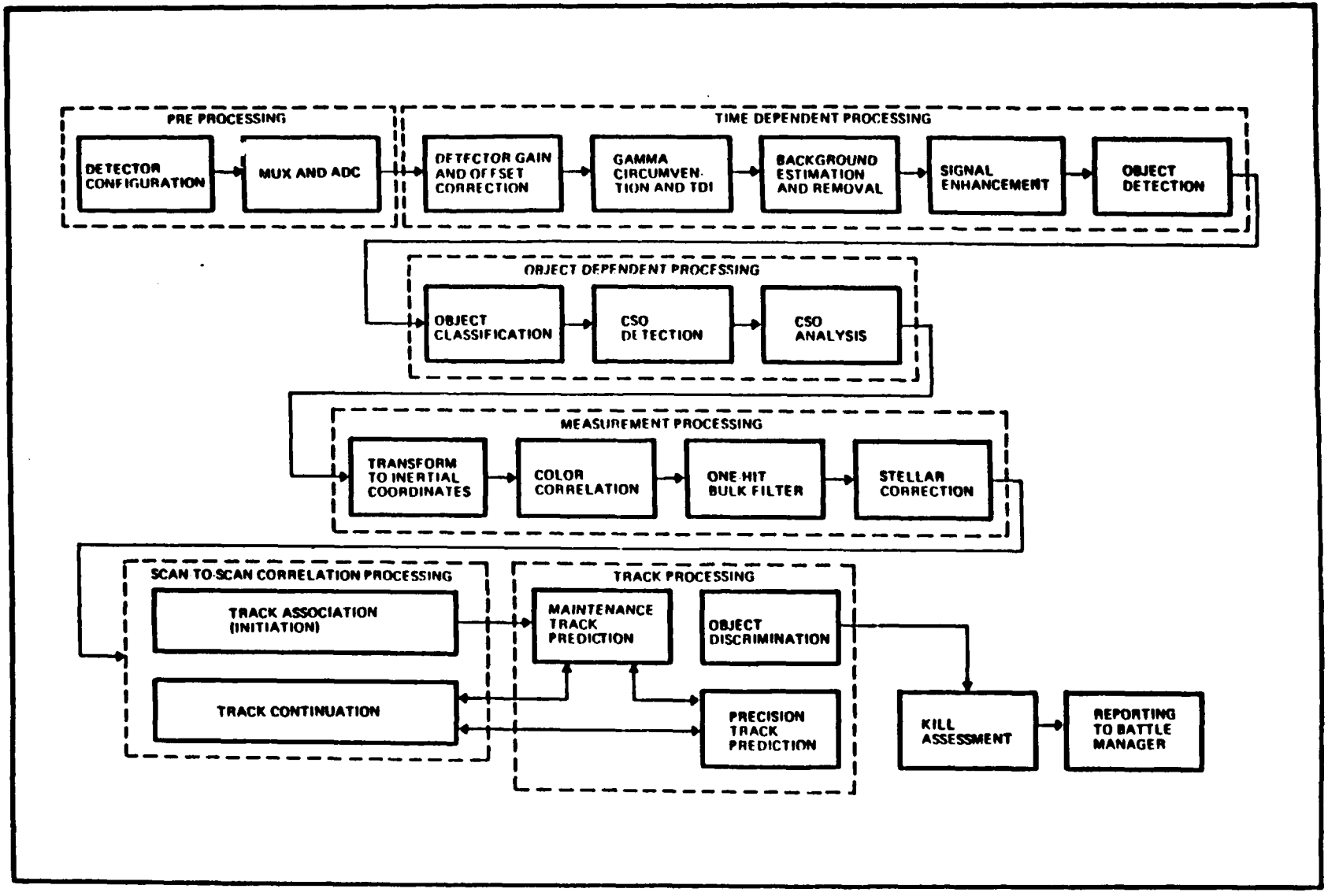


FIGURE 5-1 SDI Sensor Processing Flow Diagram (Abercrombie, 1986)

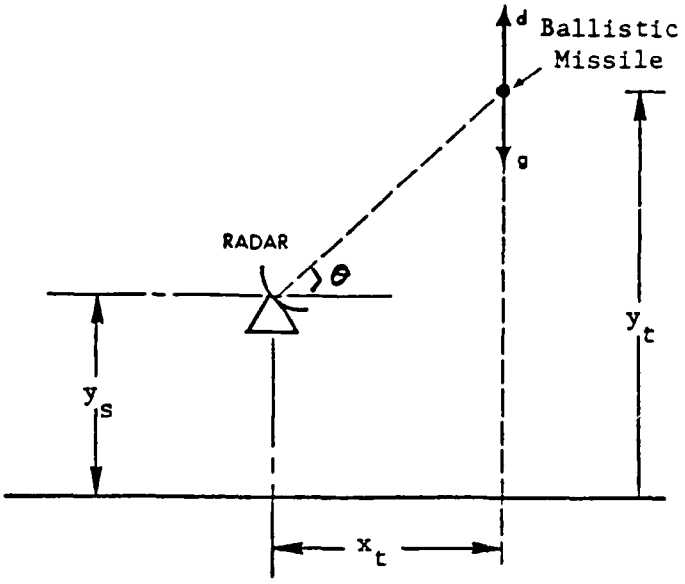


FIGURE 5-1 Geometry of a Two-Dimensional SDI Tracking Problem

TABLE 5-1 PARAMETER VALUES FOR THE ONE-DIMENSIONAL TRACKING PROBLEM

$p_o = 3.4 \times 10^{-3}$ lb sec^2/ft^4

$g = 32.2$ ft/sec^2

$k_p = 22000$ ft

$\beta \sim N(2000 \text{ lb/ft}^2, 2.5 \times 10^5 \text{lb}^2/\text{ft}^4)$

$\frac{p_o g}{2} = 0.05$ lb/ft^3

$P_{11_o} = 500$ ft^2

$P_{22_o} = 2 \times 10^4$ ft^2/sec^2

$P_{33_o} = 2.5 \times 10^5$ lb^2/ft^4

$x(0) = \hat{x}(0) = 3 \times 10^5$ ft

$\dot{x}(0) = \hat{\dot{x}}(0) = 2 \times 10^4$ ft/sec

$x_3(0) = 2 \times 10^{-2}$ ft^2/lb

$\hat{x}_3(0) = 6 \times 10^{-4}$ ft^2/lb

With angle-only measurements the line-of-sight angle, θ, can be estimated as follows:

$$\theta = \tan^{-1}\left(\frac{Y_t - Y_s}{X_t}\right) \tag{5.2}$$

The target's motion is modeled as a falling body in state variable form as:

$$X_1 = Y_t, \quad X_2 = \dot{Y}_t, \quad X_3 = \frac{1}{\beta} \tag{5.3}$$

where β is the so-called ballistic coefficient of the missile and Y_t is the target's height above the earth.

The equations of motion for the body are:

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix}}_{\dot{\underline{x}}} = \underbrace{\begin{bmatrix} x_2 \\ d-g \\ 0 \end{bmatrix}}_{\underline{f}(\underline{x})} \qquad d = \frac{\rho x_2^2}{2x_3} \tag{5.4}$$

$$\rho = \rho_0 e^{-x_1/k_\rho} \tag{5.5}$$

where d is drag deceleration, g is acceleration of gravity, ρ is atmospheric density (with ρ_0 the atmospheric density at sea level) and k_ρ is a decay constant. The differential equation for velocity, x_2, is nonlinear through the dependence of drag on velocity, air density a ballistic coefficient.

Initial values of the state variables are assumed to mean, n, and covariance matrix of the form

$$P_0 = \begin{bmatrix} P_{11_0} & 0 & 0 \\ 0 & P_{22_0} & 0 \\ 0 & 0 & P_{33_0} \end{bmatrix} \tag{5.6}$$

The problem of estimating all the state variables may be solved using an extended Kalman filter.

This SDI test problem illustrates the class of computations required for SDI target tracking. Squares, divides, square roots and trigonometric functions (sine, cosine, tangent, etc.) are needed. In addition, the extended Kalman filter requires the solution of nonlinear ordinary differential equations. Thus, high-speed nonlinear function evaluation is important to SDI track processing. This problem, although simple, can be solved relatively easily to provide a known solution to verify the parallel Kalman filter algorithms and architectures developed under this Phase I SBIR effort. This test problem can be expanded to three dimensions and angle-only measurements

of target and sensor position. In this case, the SDI track processing problem becomes more nonlinear and involves additional trig functions to be computed during coordinate transformations. The geometry for this case is discussed in the next section.

5.2 SDI PROBLEM FORMULATION AND ASPHERICAL EARTH MATH MODEL

The Kalman filter can be used to update the state estimate of a ballistic trajectory with angle only measurements. The nonlinear relationship between the state and the measurements, and the nonlinear dynamics of a ballistic trajectory, usually require the state estimate to be improved iteratively with a given measurement set. The problems due to the nonlinearities become more difficult when the observer is free-falling and more difficult still if the observer is located in the plane of the observed trajectory.

The first step in the Kalman filter equations is to project the state estimate to the time of the current measurement. This can be accomplished using the so-called f and g series, which can be found in Escobal (1975). The f and g series are derived by Taylor series expansion about the current target position. The target position at time t is given by

$$X_n = f\, X_{n-1} + g\, \dot{X}_{n-1} \tag{5.7}$$

where

X_{n-1} is the current ECI target position (x,y,z) at the time of the last measurement, t_{n-1}

$\dot{X}_{n-1}$ is the current target velocity $(\dot{x},\dot{y},\dot{z})$

$f = f\,(X_{n-1}, \dot{X}_{n-1}, t_n - t_{n-1})$

$g = g\,(X_{n-1}, \dot{X}_{n-1}, t_n - t_{n-1})$.

The target velocity at time t_n is given by

$$\dot{X}_n = \dot{f}\, X_{n-1} + \dot{g}\, \dot{X}_{n-1} \quad . \tag{5.8}$$

The f and g series given in Escobal is based on a spherical earth. Over periods of time of several hundred seconds this may introduce significant bias errors. Series approximation of the equations of motion over an aspherical earth have been derived but not implemented.

An alternative to f and g series projection of the state is numerical integration of the equations of motion. A mathematical model, given in Escobal (1975), which accounts for an aspherical earth is given in the next section.

5.2.1 SDI Target Model

$$\ddot{X} = -\frac{\mu X}{R^3}\left(1 + \frac{3}{2}\frac{J}{R^2}\left(1 - 5\left(\frac{Z}{R}\right)^2\right)\right) \tag{5.9}$$

$$\ddot{Y} = -\frac{\mu Y}{R^3}\left(1 + \frac{3}{2}\frac{J}{R^2}\left(1 - 5\left(\frac{Z}{R}\right)^2\right)\right) \tag{5.10}$$

$$\ddot{Z} = -\frac{\mu Z}{R^3}\left(1 + \frac{3}{2}\frac{J}{R^2}\left(1 - 5\left(\frac{Z}{R}\right)^2\right)\right) \tag{5.11}$$

where

μ is the gravitational constant x mass of the earth = 3.988×10^{14}

$R = (X^2 + Y^2 + Z^2)^{\frac{1}{2}}$ in meters

J is the first harmonic of the earth's gravitational potential = 4.4028×10^{10} .

5.2.2 SDI Sensor Measurement Model

$$Y = \begin{bmatrix} \theta \\ \phi \end{bmatrix} = y(X) = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \tan^{-1}\dfrac{Y_T - Y_S}{X_T - X_S} \\ \tan^{-1}\dfrac{Z_T - Z_S}{((X_T - X_S)^2 + (Y_T - Y_X)^2)^{\frac{1}{2}}} \end{bmatrix} \tag{5.12}$$

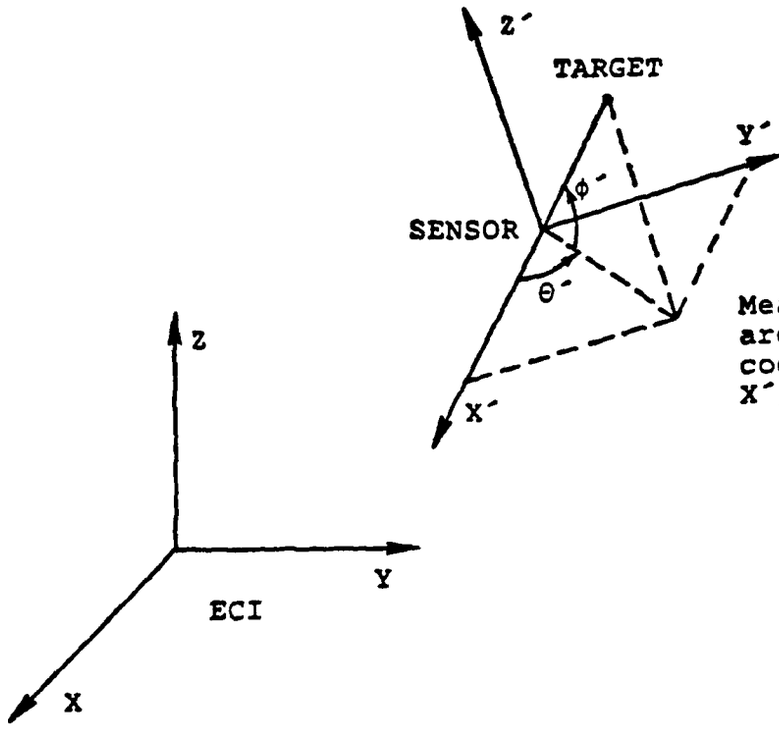
where

X_T, Y_T, Z_T is the ECI target position

X_S, Y_S, Z_S is the ECI sensor position .

In view of the problem formulation presented in this section, it is clear that SDI sensor track processing is inherently nonlinear. Squares, square roots, sines, cosines, inverse tangents and other nonlinear scalar computations characterize the target state and measurement models. These nonlinear models can be linearized about a "nominal" trajectory and a linear parallel Kalman filter used to approximate the solution.

TRANSFORMATION OF MEASUREMENTS TO ECI (Wilcox, 1986)



Measurements θ', ϕ' are made in the sensor coordinate system X´, Y´, Z´

$$\text{rotation matrix } R = \begin{bmatrix} X'_x & Y'_x & Z'_x \\ X'_y & Y'_y & Z'_y \\ X'_z & Y'_z & Z'_z \end{bmatrix}$$

where X'_y is the ECI Y component of the X´ unit vector.

$$\begin{bmatrix} X'_M \\ Y'_M \\ Z'_M \end{bmatrix} = \begin{bmatrix} \cos\theta' \cos\phi' \\ \sin\theta' \cos\phi' \\ \sin\phi' \end{bmatrix} = U'_M$$

$$U_M = \begin{bmatrix} X_M \\ Y_M \\ Z_M \end{bmatrix} = R \begin{bmatrix} X'_M \\ Y'_M \\ Z'_M \end{bmatrix}$$

$$\theta = \tan^{-1} \frac{Y_M}{X_M}$$

$$\phi = \sin^{-1} Z_M = \tan^{-1} \frac{Z_M}{(X_M^2 + Y_M^2)^{\frac{1}{2}}}$$

SECTION 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 CONCLUSIONS

Based on the results of our Phase I study, the following conclusions can be drawn:

a. It is technically feasible to decouple the predictor and corrector equations in a standard Kalman filter for parallel processing on multiple processors.

b. The decoupling principle allows the parallel Kalman filter's predictor and corrector equations to be computed on separate processors improving computational speed directly proportional to the number of available processing elements.

c. The parallel Kalman filter (PKF) is "optimal" in the same sense as the standard Kalman filter (SKF) since it was shown that the PKF is "mathematically equivalent" to the SKF (i.e., the recursive updates generated by the PKF are identical to the recursive updates of the SKF when combined in the proper fashion).

d. Parallel architectures, based on systolic array principles, have been developed that are 100% efficient when coefficient reordering is employed.

e. It is feasible to extend the linear PKF theory to nonlinear target tracking and estimation problems allowing an extended Kalman filter to run on multiple parallel processors.

In summary, it can be concluded that parallel Kalman filtering based on decoupling the filter's predictor and corrector equations is feasible. Both linear and nonlinear filtering can benefit from this unique approach. Hence, this research activity appears well suited for transition to the Phase II stage of the SBIR program.

6.2 RECOMMENDATIONS

Based on the conclusions derived from our Phase I results, the following recommendations are presented:

a. Further expand the PKF theory for nonlinear filtering and estimation. Because the SDI target tracking problem tends to be nonlinear, it is anticipated that target trajectory estimation accuracy can be substantially improved using the nonlinear equations directly.

b. Code, simulate and evaluate the PKF algorithms on a parallel computer whose architecture can be reconfigured to validate newly developed SDI algorithms and architectures. Although the PKF algorithms have been analytically shown to be optimal and stable, many issues regarding the implementation of the parallel filter can be learned by coding and simulating the PKF algorithms and architectures. For example, timing, synchronization, drift, potential divergence of the error covariance update, model sensitivities could have a major impact on the ultimate application of the PKF. Hence, it is recommended that an expert system be developed to manage PKF computations. The knowledge base of the expert system could be based on mathematically sound "rules" such as monitoring the positive definitieness of the error covariance matrix.

c. Create a Battle Management Testbed Facility (based on industry-standard hardware and software). A flexible/reconfigurable parallel processing testbed is recommended to rapidly test and evaluate the performance of newly developed SDI parallel processing algorithms and architectures. Because SDI track processing tends to be a very large nonlinear filtering problem, a scalable architecture (i.e., expandable based on problem size) for nonlinear function evaluation is recommended. General-purpose microprocessor/coprocessor technology augmented by a programmable finite state machine is recommended to accommodate a wide class of parallel algorithms. Four processors per card are recommended to simultaneously compute the equations in the decoupled PKF (i.e., two processors for the predictor and two processors for the corrector per card). Multiple cards (say eight (8)) can be installed in the testbed to validate essentially any parallel algorithm and architecture. An industry-standard Vme bus is also recommended for several reasons: 1) Vme is a very high-performance bus, 2) Vme is supported by several major companies allowing the government to add "special function" cards to the system, and 3) Vme is also standard in high-431, milspec and ruggedized systems for actual field test of our PKF technology.

d. Select a realistic SDI problem to show the benefits of the PKF technology. Because of the size and complexity of realistic SDI target tracking applications, it is anticipated that even today's supercomputer architectures will not be capable of solving these problems in near real time. Due to the unique matching of the PKF algorithms and architectures, it is anticipated that problems that could not be solved otherwise in a reasonable time (at a reasonable cost) can be solved on the proposed testbed. Thus, it is recommended that a target tracking problem of major significance to the SDI program be solved and benchmark performance documented so that future designs can be compared. Due to the

"special" architecture of the Battle Management testbed it is anticipated that it can be the standard to improve upon for the next five (5) years.

6.3 SUMMARY

Based on the results in this report it is clear that the PKF theory is well developed, mature and ready to proceed to full-scale validation on a parallel processing testbed. Because the PKF technology has been needed to solve several applications in the DoD for more than a decade, it is anticipated that once fully developed this technology can benefit several sectors of the DoD. This is possible because the necessary technology has only recently been available to transition the PKF theory into practice. Hence, Systolic Systems would be pleased to continue this program under Phase II of the SBIR program.

REFERENCES

1. Abercrombie, G.E., et. al., "Advanced Processing for Intrared Sensors", TRW Annual Report #47196-G430-029, October 1986.

2. Bucy, R.S. and K.D. Senne, "Nonlinear Filtering Algorithms for Vector Processing Machines", Compl. Appl., Vol 6, March 1980, pp. 317-338

3. Case, C.T., "DIDSIM - Defense in Depth Simulation, Sparta, Inc. Technical Report, Huntsville, AL June 1986.

4. Escobal, P.R., Methods of Orbit Determination, R.E. Krieger Publishing Co. Huntington, N.Y., 1975.

5. Gelb, A. ed., "Applied Optimal Estimation",TASC, MIT Press, Cambridge, MA, 1974 pp.194-200.

6. Jover, J.M. and T. Kailath, "A Parallel Architecture for Kalman Filter Measurement Update and Parameter Estimation"; Automation, Vol 22, No 1., pp. 43-57, 1986.

7. Meyer, G.G. and H.W. Weinert, "Parallel Algorithms and Computational Structures for Linear Estimation Problems", in Statistical Signal Processing, E.J. Wegman, ed., Marcel Dekker, Inc., 1984.

8. Miranker, W.L. and Liniger, W.M. "Parallel Methods for Integrating Ordinary Differential Equations", Math. Comp., Vol. 21, 1967, pp. 303-320.

9. Travassos, R.H., "Parallel Processing Algorithms for System Parameter Identification", Proc. IFAC Symposium on System Identification, Washington, D.C., June 1982.

10. Travassos, R.H., "VLSI Technology for the Numerical Solution of Ordinary Differential Equations," Proc SIAM 30th Aniversity Meeting, Stanford University, July 1982.

11. Travassos, R.H., and A. Andrews, "VLSI Implementation of Parallel Kalman Filters," Invited Paper, Proc. AIAA Guid. & Control Conf., Advanced Avionics Session, San Diego, Aug. 1982.

12. Travassos, R.H., "Application of Systolic Array Technology to Recursive Filter," in VLSI and Modern Signal Processing, Chapter 21, Prentice -Hall, 1983, pp. 375-388.

13. Wilcox, D.M., and D. Salazar, "Application of the Kalman Filter to the Tracking of Ballistic Trajectories with Angle Only Measurements, Nichols Research Corp. White paper, Huntsville, AL., 1986.

END

DATE

FILMED

9-88

DTIC